前　言

1989年,作者到美国新墨西哥州立大学计算机科学系做访问学者,开始学习人工神经网络。回国后,在哈尔滨工业大学为硕士研究生开设相应的选修课程。本书是在多年来所用的讲稿的基础上修改形成的。

人工神经网络的发展几经起伏,目前已有很广泛的应用。从作者开始接触该领域时的亲身感受,到了解到的中外学生初学人工神经网络所反映出的情况,我感到,开始的时候,总有一种比较神秘的感觉,加上有的资料在介绍人工神经网络时比较重视深入和全面,偏重于理论,更使得初学者在一定的时间内难以获得适当的进步,尤其是对网络基本模型的形成感到有很大的困难。为解决此问题,本书主要介绍人工神经网络的基本构成和基本的网络模型,意在使读者对人工神经网络有一个基本的了解,为他们今后从事人工神经网络的研究和应用打下一定的基础。第二个问题是人工神经网络的实验实现和应用实现问题。目前,人工神经网络有硬件实现和软件模拟两种方式。因条件所限,绝大多数初学者都是通过软件的模拟实现来体验其功能及运行特性的。而且,国内大多数的应用也是用软件实现的。因此,本书在介绍基本的人工神经网络模型的同时,还注意从软件实现的角度介绍相应的算法,甚至在最初的典型模型的介绍中,还给出了算法的具体实现。所以,本书的基本目的是:通过对人工神经网络基本构造和基本模型的介绍,使读者对其基本方法有一个基本的掌握,并能掌握如何设计出适当的计算机模拟程序,将学生引入人工神经网络及其应用的研究领域。第三是关于教育面向21世纪的问题。21世纪的科技进步、社会发展将呈现更高的速度。新世纪对学生的要求的最大不同是对其创新以及创新性地接受新技术的能力有着更高的要求。因而,除了知识的传授之外,更重要的是加强对学生创新能力的培养。总结各方面的经验,作者认为要想按照时代的要求,实现对学生"知识、能力、素质"三方面的教育,加强对学生创新能力的培养,必须重视对知识的"载体属性"的开发利用,增加教育中的理性成分。1999年下半年,作者提出了"研究型教学"的概念,希望能将现在流行的知识型教学改为"研究型教学",以便使学生建立起强烈的探索意识,培养其创新能力。对此,本人在写作中也做了探索性的尝试。在内容组织上,没有去追求知识的全面、完整,而是希望通过对一些典型网络模型的叙述,向读者介绍问题的求解方法,尤其是人工神经网络方法。虽然有多年的积累,但是"研究型教学"的概念才被提出,还需要进一步地丰富,所以这里只能说是一个非常初步的尝试。

其次,作为人工神经网络的入门,作者希望通过对人工神经网络及其基本网络模型的介绍,使学生初步了解智能系统描述的基本模型,掌握人工神经网络的基本概念、单层网、多层网、循环网等各种基本网络模型的结构、特点、典型训练算法、运行方式、典型问题、软件实现方法等。这些主要算作是"知识、能力、素质"三方面教育的知识基础部分。另外,本人还希望读者能将所学

的知识与自己未来研究课题(包括研究生论文阶段的研究课题)结合起来,通过查阅适当的参考文献,达到既丰富学习内容,又有一定的研究和应用的目的。所以,希望本书能够简明易懂、便于软件实现、鼓励研究探索。

本书分八章对人工神经网络进行介绍。第一章介绍智能的概念、智能系统的特点及其描述基本模型以及人工神经网络的特点、发展历史。第二章为人工神经网络基础,概要介绍人工神经网络的一般特性,主要包括:生物神经网络模型、人工神经元模型与典型的激励函数;人工神经网络的基本拓扑特性、存储类型及映象、训练。第三章介绍感知器与线性不可分问题、Hebb 学习律、Delta 规则。第四章介绍 BP 网络的构成及其训练过程。第五章介绍对传网的网络结构、训练。第六章介绍统计方法,主要包括:统计网络的基本训练算法、模拟退火算法与收敛分析、Cauchy 训练,人工热与临界温度在训练中的使用。第七章介绍循环网络,主要包括:循环网络的组织,稳定性分析;统计 Hopfield 网与 Boltzmann 机;基本双联存储网络的结构及训练。第八章介绍简单 ART 模型的总体结构、训练、实现。

国防科技大学的胡守仁教授审阅了原稿,提出了许多宝贵的意见和建议,在此对胡先生认真负责的精神以及对作者的帮助和爱护表示真诚的谢意!由于作者水平有限,书中错误和不当之处在所难免,敬请读者批评指正。

编者

2001.2

目　录

第一章　引言 …………………………………… (1)
1.1　人工神经网络的提出 ………………… (1)
1.1.1　智能与人工智能 ………………… (1)
1.1.2　物理符号系统 …………………… (4)
1.1.3　联接主义观点 …………………… (5)
1.1.4　两种模型的比较 ………………… (6)
1.2　人工神经网络的特点 ………………… (7)
1.2.1　人工神经网络的概念 …………… (7)
1.2.2　学习能力 ………………………… (8)
1.2.3　普化能力 ………………………… (8)
1.2.4　信息的分布存放 ……………… (10)
1.2.5　适用性问题 …………………… (10)
1.3　历史回顾 …………………………… (10)
1.3.1　萌芽期 ………………………… (10)
1.3.2　第一高潮期 …………………… (11)
1.3.3　反思期 ………………………… (11)
1.3.4　第二高潮期 …………………… (12)
1.3.5　再认识与应用研究期 ………… (13)
练习题 …………………………………… (13)
第二章　人工神经网络基础 …………………… (15)
2.1　生物神经网络 ……………………… (15)
2.2　人工神经元 ………………………… (16)
2.2.1　人工神经元的基本构成 ……… (16)
2.2.2　激活函数(Activation Function) …… ……………………………………………… (17)
2.2.3　M－P模型 ……………………… (19)
2.3　人工神经网络的拓扑特性 ………… (19)
2.3.1　联接模式 ……………………… (20)
2.3.2　网络的分层结构 ……………… (20)
2.4　存储与映射 ………………………… (25)
2.5　人工神经网络的训练 ……………… (27)
2.5.1　无导师学习 …………………… (27)
2.5.2　有导师学习 …………………… (28)
练习题 …………………………………… (29)
第三章　感知器 ……………………………… (30)
3.1　感知器与人工神经网络的早期发展 ………………………………… (30)
3.2　感知器的学习算法 ………………… (31)
3.2.1　离散单输出感知器训练算法 … (31)
3.2.2　离散多输出感知器训练算法 … (32)
3.2.3　连续多输出感知器训练算法 … (33)
3.3　线性不可分问题 …………………… (35)
3.3.1　异或(Exclusive－OR)问题 …… (35)
3.3.2　线性不可分问题的克服 ……… (37)
练习题 …………………………………… (38)
第四章　BP 网络 ……………………………… (39)
4.1　概述 ………………………………… (39)
4.2　基本 BP 算法 ……………………… (40)
4.2.1　网络的构成 …………………… (40)
4.2.2　训练过程概述 ………………… (41)
4.2.3　误差传播分析 ………………… (42)
4.2.4　基本的 BP 算法 ……………… (44)
4.3　算法的改进 ………………………… (45)
4.4　算法的实现 ………………………… (47)
4.5　算法的理论基础 …………………… (48)
4.6　几个问题的讨论 …………………… (52)
练习题 …………………………………… (54)
第五章　对传网 ……………………………… (55)
5.1　网络结构 …………………………… (56)
5.2　网络的正常运行 …………………… (57)
5.2.1　Kohonen 层 …………………… (57)
5.2.2　Grossberg 层 ………………… (58)

5.3 Kohonen 层的训练 …………………… (59)
5.3.1 输入向量的预处理 …………… (59)
5.3.2 训练 ……………………………… (60)
5.4 Kohonen 层联接权的初始化方法 … (62)
5.5 Grossberg 层的训练 ………………… (65)
5.6 补充说明 ……………………………… (67)
练习题 …………………………………………… (69)
第六章 非确定方法 ………………………… (70)
6.1 基本的非确定训练算法 …………… (70)
6.2 模拟退火算法 ……………………… (73)
6.3 Cauchy 训练 ………………………… (78)
6.4 相关的几个问题 …………………… (79)
练习题 …………………………………………… (82)
第七章 循环网络 …………………………… (83)
7.1 循环网络的组织 …………………… (83)
7.2 稳定性分析 ………………………… (87)
7.3 统计 Hopfield 网与 Boltzmann 机 …… (90)
7.4 双联存储器的结构 ………………… (94)
7.5 异相联存储 ………………………… (96)
7.6 其他的双联存储器 ………………… (97)
7.7 Hopfield 网用于解决 TSP 问题 …… (97)
练习题 …………………………………………… (100)
第八章 自适应共振理论 …………………… (101)
8.1 ART 的结构 ………………………… (102)
8.2 ART 的初始化 ……………………… (107)
8.2.1 T 的初始化 ……………………… (107)
8.2.2 B 的初始化 ……………………… (107)
8.2.3 ρ 的初始化 ……………………… (108)
8.3 ART 的实现 ………………………… (108)
练习题 …………………………………………… (112)
参考文献 ……………………………………… (114)

第一章 引 言

早在电子计算机出现之前，人类就已经开始探索智能的秘密，并且期盼着有一天可以重新构造人脑，让其去代替人类完成相应的工作。这种目标一直鼓励着人们不断地努力。大体上讲，人类对人工智能的研究可以分成两种方式，这两种方式对应着两种不同的技术：传统的人工智能技术和基于人工神经网络的技术。实际上，这两种技术是分别从心理的角度和生理的角度对智能进行模拟的。因此，它们分别适应于认识和处理事物(务)的不同方面。目前，人们除了分别从不同的角度对这两种技术进行研究外，也已开始探讨如何能将这两种技术更好地结合起来，并且已取得了良好的效果。人们期待着，通过大家的不懈努力，在不久的将来，能在这两种技术的研究上以及它们的有机结合方面有所突破，也希望在方法上有一个新的突破，真正打开智能的大门。

本章首先简要介绍智能和人工智能，然后简要介绍人工神经网络的基本特点及其发展过程，使读者对有关的概念有一个基本的了解。

1.1 人工神经网络的提出

人工神经网络(Artificial Neural Networks，简记作 ANN)，是对人类大脑系统的一阶特性的一种描述。简单地讲，它是一个数学模型，可以用电子线路来实现，也可以用计算机程序来模拟，是人工智能研究的一种方法。因此，需要先介绍人工智能的一些基本内容。

1.1.1 智能与人工智能

一、智能的含义

众所周知，人类是具有智能的。因为人类能记忆事物，能有目的地进行一些活动，能通过学习获得知识，并能在后续的学习中不断地丰富知识，还有一定的能力运用这些知识去探索未知的东西，去发现、去创新。那么，智能的含义究竟是什么？如何刻划它呢？

粗略地讲，**智能是个体有目的的行为，合理的思维，以及有效的适应环境的综合能力**。也可以说，**智能是个体认识客观事物和运用知识解决问题的能力**。

按照上述描述，人类个体的智能是一种综合能力。具体来讲，可以包含如下八个方面的能力：

1. 感知与认识客观事物、客观世界和自我的能力

这是人类在自然界中生存的最基本的能力，是认识世界、推动社会发展的基础。人类首先必须感知客观世界，使客观世界中的事物在自己的头脑中有一个反映，并根据事物反映出来的不同特性将事物区分开来。这是一切活动的基础。“假物必以用者”，只有认识了事物，我们才能制造

出支持生存、生活的工具，才有可能不断地提高人类的生存能力，并不断地改善人类的生活质量。

因此可以说，感知是智能的基础。

2. 通过学习取得经验与积累知识的能力

这是人类在自然界中能够不断发展的最基本的能力。通过学习不断地取得经验、不断地积累知识，又进一步地增强了人类认识客观事物、客观世界和自我的能力，从而推动人类社会不断发展。而且，随着社会的发展，知识的积累不仅孤立地发生在作为个体的人的身上，更重要的是这种积累能够代代相传。先辈们获取的经验、知识通过一定的形式传给下一代。正是这样，才使得人类所掌握的知识越来越多，越来越丰富，以至于人们称现在是知识爆炸的时代。这表明，随着社会的进步，人类的知识积累速度不断加快。

3. 理解知识，运用知识和经验分析、解决问题的能力

这一能力可以算作是智能的高级形式，是人类对世界进行适当的改造、推动社会不断发展的基本能力。

有了知识以后，要使其发挥作用，必须运用这些知识和经验去分析和解决实际问题。所以，作为教育的重要目标，我们要努力培养学生认识问题、分析问题和解决问题的能力。培根说，知识就是力量。他指的是，当知识得到恰当的应用后，会发挥巨大的作用。所以，大师们一直在告诫人们，不要只读书，尤其不要死读书，要在灵活地运用书本上的知识去解决实际问题、并在应用中不断地丰富知识上下功夫。

4. 联想、推理、判断、决策的能力

这是智能的高级形式的又一方面。

人类通过这种能力，去促进对未来的甚至是未知的东西的预测和认识，使我们具有了一定的判断未来、把握未来的能力，使我们对未来的东西也能有所准备，从而进一步增强了我们在这个世界上生存并不断发展的能力。我们说，无论是学习、工作还是生活，都有"主动"和"被动"之分。联想、推理、判断、决策的能力是"主动"的基础；同时，它也是我们有时要"主动"地采用"被动"策略去更有效地解决问题的基础——因为我们较好地掌握了事物发展的趋势。

5. 运用语言进行抽象、概括的能力

人类的语言是最为丰富的，它除了可以表达实际世界中的事物外，还可以表达出人类的情感以及一些直观不可见的东西，这些使得我们的生活更加丰富多彩。抽象和概括已成为人类认识现实世界和未来世界的一个重要工具。从更高的形式来看，它是形式化描述的基础，而形式化描述则是计算机化、自动化的基础。

正是有了语言，人类才有了交流，而且这种交流被广泛地扩展到了人与机器之间，使得机器能更好地完成人类所交付的各项任务。丰富的语言抽象和概括能力，使得其他方面的能力可以更充分地被发挥出来。

上述这五种能力，被认为是人类智能最基本的能力，从一定的意义上讲，后续的三种能力是这五种能力的新的综合表现形式。

6. 发现、发明、创造、创新的能力

这种能力主要是前面的第三种能力的一种高级表现形式，在这里，我们强调的更多的是创新

能力。因为只有创新，才能有活力，才能不断地发展。人类正是在不断地有所发明、有所创造中前进的。

7. 实时、迅速、合理地应付复杂环境的能力

这是实时反应能力。表示人类对自己遇到的环境及事务可以做出适当的反应。因为世界上几乎所有的事务都将时间作为一个"自变量"而随其变化而变化，人类面对繁乱复杂的环境，必须有能力做出"实时"恰当的反映。从一定的意义上说，这也是人类生存的基本能力。

8. 预测、洞察事物发展、变化的能力

根据历史的经验，根据现实的信息，判断事物的未来发展，以对未来将出现的事务做出必要的准备。

那么，如何使类似计算机这样的设备去模拟人类的这些能力呢？这就是人工智能所研究的问题。

二、人工智能

人工智能(Artificial Intelligence，简记为AI)最初是在1956年被引入的。它研究怎样让计算机模仿人脑从事推理、设计、思考、学习等思维活动，以解决和处理较复杂的问题。简单地来说，人工智能就是研究如何让计算机模仿人脑进行工作。

可以将研究人工智能的目的归纳为两个方面：

1. 增加人类探索世界、推动社会前进的能力

人类从一开始就注意到通过制造和使用工具来加强和延伸自己的生存能力。在初始阶段，这种工具多是用于扩展人类的体力，例如杠杆、拖拉机、各类机械等，它们主要致力于放大、聚集、集中、产生"力量"。后来，从考虑如何让"工具"帮助人类完成计算开始，人们致力于研究如何能使"工具"代替人类进行思维，哪怕是"部分地"代替！当计算机出现之后，更进一步地促使人类探索如何使计算机模拟人的感知、思维和行为的规律，进而设计出具有类似人类的某些智能的计算机系统，从而达到延伸和扩展人类智能和能力的目的。

2. 进一步认识自己

到目前为止，虽然以生物神经科学家、医学解剖学家为首的各专业的科学家进行了数代的大量艰苦的研究，人类对自身的大脑还是知之甚少，在很大程度上，它的运行机理还是一个未揭开的谜。研究人工智能，可以从已知的一些结论(不排除一些猜想)入手，从人的大脑以外来探讨它的活动机理。有人将这种做法叫做用物化了的智能去考察和研究人脑智能的物质过程和规律。这也许是人类揭开自身大脑之谜的一个有效途径。人们相信，这种探索至少可以为我们认识大脑提供帮助。

由于人类对自己的大脑确实知之甚少，所以，自从"人工智能"一词诞生以来，人们从不同的出发点、方法学以及不同的应用领域出发，进行了大量的研究。正是由于存在这些不同，导致了对人工智能的几种不同的认识，也就形成了不同的学术流派。较有代表性的包括：符号主义(或叫做符号/逻辑主义)学派，联接主义(或者叫做并行分布处理)学派，进化主义(或者叫做行动/响应)学派。

1.1.2 物理符号系统

人们常讲,计算机世界就是数据处理世界,而数据是从现实世界中抽象出来的信息世界的形式化描述的结果。当然,这种形式化系统的不同,会导致数据世界的不同。所以,信息是现实在人脑中的反映,而数据则是信息的一种表现形式,如图 1-1 所示。信息不会随其载体的变化而变化,而数据则是随其载体的变化而变化的。例如,“2”在十进制中用阿拉伯数字表示成“2”,而在二进制中又被表示成“10”,在计算机内部,它又被用高、低电平表示出来。因此,信息需要在一定的载体上以某种规定的形式表达出来。习惯上,人们用一系列的基本符号以及组合这些符号的一些规则去表达一些信息和行为。这些基本符号以及组合这些符号的规则就是所谓的物理符号系统。

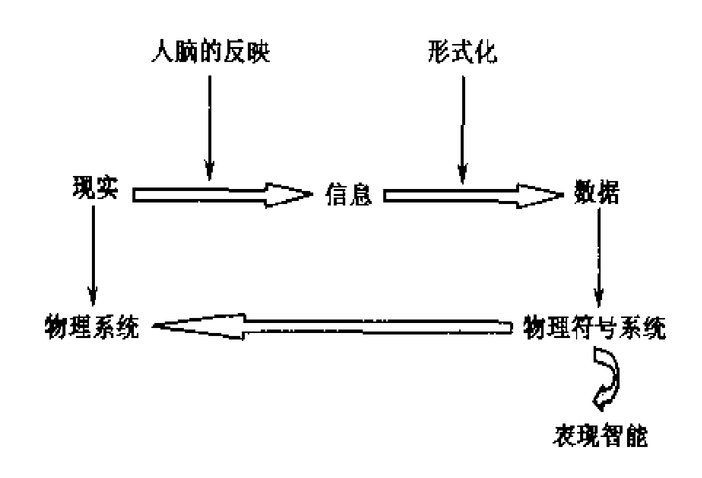


图 1-1 物理符号系统用于对物理系统的描述

物理符号系统是 Newell 和 Simon 在 1967 年提出的假说,该假说认为:

一个物理系统表现智能行为的充要条件是它有一个物理符号系统。

这就是说,物理符号系统需要有一组称为符号的实体组成,它们都是物理模型,可以在另一类称为符号结构的实体中作为成分出现,以构成更高级别的系统。

在这里,人们希望通过抽象,用一系列物理符号及其相应的组成规则,来表达一个物理系统的存在和运行。例如简单的整数及其运算系统、实数及其运算系统、数理逻辑符号系统。传统的人工智能技术就是以物理符号系统为基础的。在这里,问题必须经过形式化处理后才能被表达、处理。

要想实现对事务(物)的形式化描述,第一步必须对其进行适当的抽象。然而我们知道,在抽象中,需要舍弃一些特性,同时保留一些特性。但是,世界的千差万别要求物理符号系统能较好地去表达我们要求的全部,在一定意义上讲,这与抽象又存在一定的矛盾。因为,为了形式化所进行的抽象有时需要舍弃大量的信息,而这将导致经过形式化处理后的系统难以表达出物理系统的完整面貌。更严重的是,有时还会使其失去物理系统的本来面貌。在现实世界中,这种问题有许多。实际上,在某些情况下如果我们勉强对它们进行形式化处理,一方面会导致面目全非,另一方面可能会因为过于复杂等问题,使得系统难以具有良好的结构。我们称此类问题是难以

形式化的。这是物理符号系统所面临的困难。

由此也可以看出，物理符号系统对全局性判断、模糊信息处理、多粒度的视觉信息处理等是非常困难的，这就导致了人们去探求对此类问题的新的处理方法。

1.1.3 联接主义观点

为了研究智能，在现代神经科学的研究成果的基础上，人们提出了另一种观点，认为：

智能的本质是联接机制。神经网络是一个由大量简单的处理单元组成的高度复杂的大规模非线性自适应系统。

虽然按此说法来刻划神经网络，未能将其所有的特性完全描述出来，但它却从以下四个方面出发，力图最大限度地体现人脑的一些基本特征，同时使得所得人工神经网络具有良好的可实现性。人工神经网络就是力求从这四个方面去模拟人脑的智能行为。

1. 物理结构

现代神经科学的研究结果认为，大脑皮层是一个广泛联接的巨型复杂系统，它包含有大约一千亿个神经元，这些神经元通过一千万亿个联接构成一个大规模的神经网络系统。人工神经网络也将是由与生物神经元类似的人工神经元通过广泛的联接构成的。人工神经元将模拟生物神经元的功能。它们不仅具有一定的局部处理能力，同时还可以接受来自系统中其他神经元的信号，并可以将自己的“状态”按照一定的形式和方式传送给其他的神经元。

2. 计算模拟

人脑中的神经元，既有局部的计算和存储功能，又通过联接构成一个统一的系统。人脑的计算就是建立在这个系统的大规模并行模拟处理的基础上的。各个神经元可以接受系统中其他神经元通过联接传送过来的信号，通过局部的处理，产生一个结果，再通过联接将此结果发送出去。神经元接受和传送的信号被认为是模拟信号。所有这些，对大脑中的各个神经元来说，都是同时进行的。因此，该系统是一个大规模并行模拟处理系统。由于人工神经网络中存在大量的有局部处理能力的人工神经元，所以，该系统也将实现信息的大规模并行处理，以提高其性能。

3. 存储与操作

研究认为，大脑对信息的记忆是通过改变突触(Synapse)的联接强度来实现的。神经元之间的联接强度确定了它们之间传递的信号的强弱，而联接强度则由相应的突触决定。也就是说，除神经元的状态所表现出的信息外，其他信息被以神经元之间联接强度的形式分布存放。存储区与操作区合二为一。这里的处理是按大规模、连续、模拟方式进行的。由于其信息是由神经元的状态和神经元之间实现联接的突触的强弱所表达的，所以说信息的分布存放是它的另一个特点。这是人工神经网络模拟实现生物神经系统的第三大特点。

信息的大规模分布存放给信息的充分并行处理提供了良好的基础。同时，这些特性又使系统具有了较强的容错能力和联想能力，也给概括、类比、推广提供了强有力的支持。

4. 训练

生活实践的经验告诉我们，人的大脑的功能除了受到先天因素的限制外，还被后天的训练所确定。先天因素和后天因素中，后天的训练更为重要。一个人的学习经历、工作经历都是他的宝

贵财富。这些表明，人脑具有很强的自组织和自适应性。同我们看到的表象不同，从生理的角度来讲，人的许多智力活动并不是按逻辑方式进行的，而是通过训练形成的。所以，人工神经网络将根据自己的结构特性，使用不同的训练、学习过程，自动从"实践"中获取相关的知识，并将其存放在系统内。这里的"实践"就是训练样本。

实际上，虽然人类对神经网络的研究起源很早，然而真正广泛地将其用作人工智能的一项新的技术来研究只是近几十年的事。这种努力在20世纪60年代受到挫折后，停顿了近20年。后来，人们发现传统的人工智能技术要在近期取得大的突破还较为困难，同时加上人们在生物神经网络和人工神经网络方面研究的进展，重新唤起了人们对用人工神经网络来实现人工智能的兴趣。希望通过共同的努力，尽快构造出一个较为理想的人工智能系统。

为此，许多方面的科学家分别从各自的学科入手，交叉联合，进行研究。所以说，人工神经网络理论是许多学科共同努力的结果。这些学科主要包括神经科学、生物学、计算机科学与技术、生理学、数学、工程、心理学、哲学、语言学等。

1.1.4 两种模型的比较

物理符号系统和人工神经网络系统从不同的方面对人脑进行模拟。其差别见表1-1。

表1-1 物理符号系统和人工神经网络系统的差别

项目	物理符号系统	人工神经网络
处理方式	逻辑运算	模拟运算
执行方式	串行	并行
动作	离散	连续
存储	局部集中	全局分布

可以说，物理符号系统是从人的心理学的特性出发，去模拟人类问题求解的心理过程。所以它擅长于模拟人的逻辑思维，可以将它看作是思维的高级形式。而在许多系统中，一些形象思维的处理需要用逻辑思维来实现，这就导致了该系统对图像处理类问题的处理效率不高。

作为联接主义观点的人工神经网络，它是从仿生学的观点出发，从生理模拟的角度去研究人的思维与智能，擅长于对人的形象思维的模拟，这是人类思维的低级形式。从目前的研究结果看，因为这种系统的非精确性的特点，使得它对以逻辑思维为主进行求解的问题的处理较为困难。图1-2给出了两种系统与人类思维形式的对应比较。

这两种观点导致了两种不同的人工智能技术：基于物理符号系统的传统的人工智能技术和基于联接主义观点的人工神经网络技术。这两种技术的比较见表1-2。从表中可以看出，这两种技术导致处理问题方法的不同，使得相应系统的开发方法和适应的对象有着很大的差别。按照这一分析，传统的人工智能方法和人工神经网络的方法并不是完全可以互相取代的，它们应该有着不同的应用面。

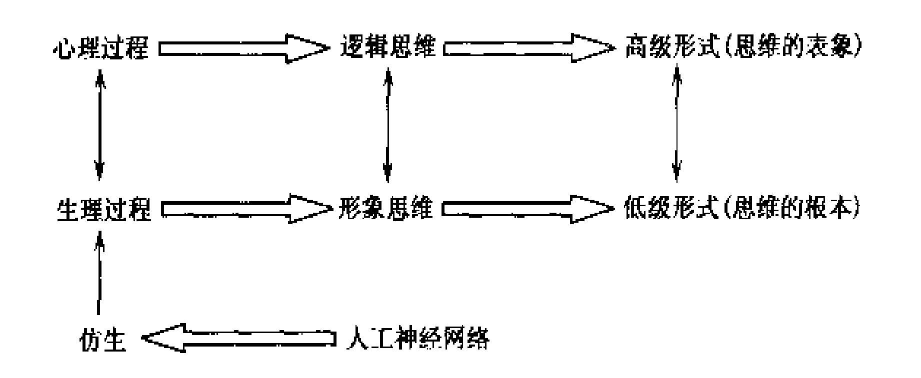


图 1－2　两种模型的模拟对照

表 1－2　两种人工智能技术的比较

项目	基于物理符号系统的传统的人工智能技术	基于联接主义观点的人工神经网络技术
基本实现方式	串行处理;由程序实现控制	并行处理;对样本数据进行多目标学习;通过人工神经元之间的相互作用实现控制
基本开发方法	设计规则、框架、程序;用样本数据进行调试(由人根据已知的环境去构造一个模型)	定义人工神经网络的结构原型,通过样本数据,依据基本的学习算法完成学习——自动从样本数据中抽取内涵(自动适应应用环境)
适应领域	精确计算:符号处理,数值计算	非精确计算:模拟处理,感觉,大规模数据并行处理
模拟对象	左脑(逻辑思维)	右脑(形象思维)

1.2　人工神经网络的特点

信息的分布表示、运算的全局并行和局部操作、处理的非线性是人工神经网络的三大特点。其构造和处理均是围绕此三点进行的。

1.2.1　人工神经网络的概念

1. 定义

人工神经网络是人脑及其活动的一个理论化的数学模型,它由大量的处理单元通过适当的方式互连构成,是一个大规模的非线性自适应系统。1988 年,Hecht—Nielsen 曾经给人工神经网络下了如下的定义:

人工神经网络是一个并行、分布处理结构,它由处理单元及称为联接的无向信号通道互连而成。这些处理单元(PE——Processing Element)具有局部内存,并可以完成局部操作。每个处理单元有一个单一的输出联接,这个输出可以根据需要被分支成希望个数的许多并行联接,且这些并行联接都输出相同的信号,即相应处理单元的信号,信号的大小不因分支的多少而变化。处理单元的输出信号可以是任何需要的数学模型,每个处理单元中进行的操作必须是完全局部的。

也就是说，它必须仅仅依赖于经过输入联接到达处理单元的所有输入信号的当前值和存储在处理单元局部内存中的值。

该定义主要强调了四个方面的内容：并行、分布处理结构；一个处理单元的输出可以被任意分支，且大小不变；输出信号可以是任意的数学模型；处理单元完全的局部操作。这里说的处理单元就是人工神经元（AN——Artificial Neuron）。

按照 Rumellhart、McClelland、Hinton 等人提出的 PDP（Parallel Distributed Processing）理论框架（简称为 PDP 模型），人工神经网络由八个方面的要素组成：

(1) 一组处理单元（PE 或 AN）；

(2) 处理单元的激活状态（a_i）；

(3) 每个处理单元的输出函数（f_i）；

(4) 处理单元之间的联接模式；

(5) 传递规则（$\sum w_{ij}O_i$）；

(6) 把处理单元的输入及当前状态结合起来产生激活值的激活规则（F_i）；

(7) 通过经验修改联接强度的学习规则；

(8) 系统运行的环境（样本集合）。

可以将 PDP 模型表示成图 1－3 的形式。

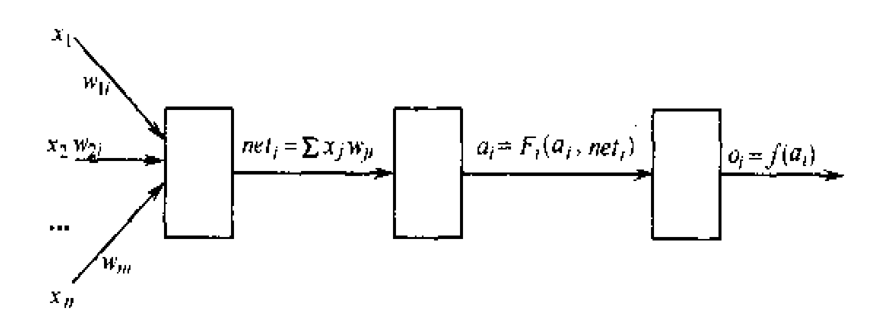


图 1－3　PDP 模型下的人工神经元网络模型

以上这两种定义都比较详细、复杂。为了使用方便，1987 年，Simpson 从人工神经网络的拓扑结构出发，给出了一个虽然不太严格但却是简明扼要的定义。它对一般的应用来说，是足以说明问题的：

人工神经网络是一个非线性的有向图，图中含有可以通过改变权大小来存放模式的加权边，并且可以从不完整的或未知的输入找到模式。

人工神经网络除了可以叫做并行分布处理系统（PDP）外，还可以叫做人工神经系统（ANS）、神经网络（NN）、自适应系统（Adaptive Systems）、自适应网（Adaptive Networks）、联接模型（Connectionism）、神经计算机（Neurocomputer）等。

人工神经网络不仅在形式上模拟了生物神经系统，它也确实具有大脑的一些基本特征：

(1) 神经元及其联接

从系统构成的形式上看，由于人工神经网络是受生物神经系统的启发构成的，从神经元本身

到联接模式，基本上都是以与生物神经系统相似的方式工作的。这里的人工神经元(AN)与生物神经元(BN)相对应，可以改变强度的联接则与突触相对应。

(2) 信息的存储与处理

从表现特征上来看，人工神经网络也力求模拟生物神经系统的基本运行方式。例如，可以通过相应的学习/训练算法，将蕴含在一个较大数据集中的数据联系抽象出来。就像人们可以不断地摸索规律、总结经验一样，可以从先前得到的例子按要求产生出新的实例，在一定程度上实现"举一反三"的功能。

1.2.2 学习能力

人工神经网络可以根据所在的环境去改变它的行为。也就是说，人工神经网络可以接受用户提交的样本集合，依照系统给定的算法，不断地修正用来确定系统行为的神经元之间联接的强度，而且在网络的基本构成确定之后，这种改变是根据其接受的样本集合自然地进行的。一般来说，用户不需要再根据所遇到的样本集合去对网络的学习算法做相应的调整。也就是说，人工神经网络具有良好的学习功能。由于在传统的人工智能系统的研究中，虽然人们对"机器学习"问题给予了足够的重视并倾注了极大的努力，但是，系统的自学习能力差依然是阻碍其获得广泛应用的最大障碍。而人工神经网络具有良好的学习功能的这一性能，使得人们对它产生了极大的兴趣。人工神经网络的这一特性称为"自然具有的学习功能"，以与传统的人工智能系统总要花较大的力气去研究系统的学习问题形成对照。

在学习过程中，人工神经网络不断地从所接受的样本集合中提取该集合所蕴含的基本东西，并将其以神经元之间的联接权重的形式存放于系统中。例如，可以构造一个异相联的网络，它在接受样本集合 $\boldsymbol{A}$ 时，可以抽取集合 $\boldsymbol{A}$ 中输入数据与输出数据之间的映射关系。如果样本集合变成了 $\boldsymbol{B}$，它同样可以抽取集合 $\boldsymbol{B}$ 中输入数据与输出数据之间的映射关系。再例如，对于某一模式，可以用它的含有不同噪声的数据去训练一个网络，在这些数据选择得比较恰当的前提下，可以使得网络今后在遇到类似的含有一定缺陷的数据时，仍然能够得到它对应的完整的模式。也可以说，这表明，人工神经网络可以学会按要求产生它从未遇到过的模式。有时候，又将人工神经网络的这一功能叫做"抽象"功能。

目前，对应不同的人工神经网络模型，有不同的学习/训练算法，有时，同种结构的网络拥有不同的算法，以适应不同的应用要求。对一个网络模型来说，其学习/训练算法是非常重要的。例如，作为一般的多级网络学习/训练算法的 BP 算法，虽然已被发现并应用多年，今天仍然有许多人在研究如何提高它的训练速度和性能。

1.2.3 普化能力

由于其运算的不精确性，人工神经网络在被训练后，对输入的微小变化是不反应的。与事物的两面性相对应，虽然在要求高精度计算时，这种不精确性是一个缺陷，但是，有些场合又可以利用这一点获取系统的良好性能。例如，可以使这种不精确性表现成"去噪音、容残缺"的能力，而这对模式识别有时恰好是非常重要的。还可以利用这种不精确性，比较自然地实现模式的自动

分类。

尤其值得注意的是，人工神经网络的这种特性不是通过隐含在专门设计的计算机程序中的人类的智能来实现的，而是其自身的结构所固有的特性所给定的。

1.2.4 信息的分布存放

信息的分布存放给人工神经网络提供了另一种特殊的功能。由于一个信息被分布存放在几乎整个网络中，所以，当其中的某一个点或者某几个点被破坏时，信息仍然可以被存取。这能够保证系统在受到一定的损伤时还可以正常工作。但是，这并不是说，可以任意地对完成学习的网络进行修改。也正是由于信息的分布存放，对一类网来说，当它完成学习后，如果再让它学习新的东西，这时就会破坏原来已学会的东西，BP 网就是这类网。有关具体分析见相关章节。

1.2.5 适用性问题

从 1.1.4 节可以看出，人工神经网络并不是可以解决所有问题的，它应该有自己的适用面。人脑既能进行“形象思维”又能进行“逻辑思维”，传统的人工智能技术模拟的是逻辑思维，人工神经网络模拟的是形象思维，而这两者适用的方面是不同的，所以，人工神经网络擅长于适用形象思维的问题的处理。主要包括两个方面：

(1) 对大量的数据进行分类，并且只有较少的几种情况；

(2) 必须学习一个复杂的非线性映射。

这两个方面对传统的人工智能技术来说都是比较困难的。目前，人们主要将其用于语音、视觉、知识处理、辅助决策等方面。此外，在数据压缩、模式匹配、系统建模、模糊控制、求组合优化问题的最佳解的近似解(不是最佳近似解)等方面也有较好的应用。

1.3 历史回顾

人工神经网络的发展是曲折的，从萌芽期到目前，几经兴衰。可以将其发展历史大体上分成如下五个时期。

1.3.1 萌芽期

人工神经网络的研究最早可以追溯到人类开始研究自己的智能的时期，这一时期截止到 1949 年。

开始时，人类对自身的思维感到非常奇妙，从而也就有了许许多多关于思维的推测，这些推测既有解剖学方面的，也有精神方面的。一直到了神经解剖学家和神经生理学家提出人脑的“通信联接”机制，我们才对人脑有了一点了解。到了 20 世纪 40 年代初期，对神经元的功能及其功能模式的研究结果才足以使研究人员通过建立起一个数学模型来检验他们提出的各种猜想。在这个期间，产生了两个重大成果，它们构成了人工神经网络萌芽期的标志。

1943 年，心理学家 McCulloch 和数学家 Pitts 建立起了著名的阈值加权和模型，简称为 M－P

模型。1943年,McCulloch和Pitts总结了生物神经元的一些基本生理特征,对其一阶特性进行形式化描述,提出了一种简单的数学模型与构造方法,这一结果发表在数学生物物理学会刊《Bulletin of Methematical Biophysics》上。这为人们用元器件、用计算机程序实现人工神经网络打下了坚实的基础。

1949年,心理学家D.O.Hebb提出神经元之间突触联系是可变的假说。他认为,人类的学习过程是发生在突触上的,而突触的联接强度则与神经元的活动有关。据此,他给出了人工神经网络的学习律——联接两个神经元的突触的强度按如下规则变化:在任意时刻,当这两个神经元处于同一种状态时,表明这两个神经元具有对问题响应的一致性,所以,它们应该互相支持,其间的信号传输应该加强,这是通过加强它们之间的突触的联接强度实现的。反之,在某一时刻,当这两个神经元处于不同的状态时,表明它们对问题的响应是不一致的,因此它们之间的突触的联接强度被减弱。称之为Hebb学习律。Hebb学习律在人工神经网络的发展史中占有重要的地位,被认为是人工神经网络学习训练算法的起点,是里程碑。

1.3.2 第一高潮期

第一高潮期大体上可以认为是从1950年到1968年,也就是从单级感知器(Perceptron)的构造成功开始,到单级感知器被无情地否定为止。这是人工神经网络的研究被广为重视的一个时期。其重要成果是单级感知器及其电子线路模拟。

在20世纪50年代和60年代,一些研究者把生理学和心理学的观点结合起来,研究成功了单级感知器,并用电子线路去实现它。电子计算机出现后,人们才转到用更方便的电子计算机程序去模拟它。由于用程序进行模拟既便于修改又便于测试,而且更重要的是,这种方法的费用特别低。所以,直到今天,大批的甚至是大多数的研究人员仍然在用这种模拟的方法进行研究。本书在后面的叙述中,也将对人工神经网络的程序模拟方法进行适当的讨论。

这个期间的研究以Marvin Minsky、Frank Rosenblatt、Bernard Widrow等为代表人物,代表作是单级感知器。它被人们用于各种问题的求解,甚至在一段时间里,它使人们乐观地认为几乎已经找到了智能的关键。

早期的成功,给人们带来了极大的兴奋。不少人认为,只要其他的技术条件成熟,就可以重构人脑,因为重构人脑的问题已转换成建立一个足够大的网络的问题。包括美国政府在内的许多部门都开始大批地投入此项研究,希望尽快占领制高点。

1.3.3 反思期

正在人们兴奋不已的时候,M. L. Minsky和S. Papert对单级感知器进行了深入的研究,从理论上证明了当时的单级感知器无法解决许多简单的问题。在这些问题中,甚至包括最基本的"异或"问题。这一成果在《Perceptron》一书中发表,该书由MIT出版社在1969年出版发行。以该书的出版为标志,人们对人工神经网络的研究进入了反思期。

由于"异或"运算是计算机中的最基本运算之一,所以,这一结果是令人震惊的。由于Minsky的卓越、严谨和威望,使得不少人对此结果深信不疑。从而导致了许多研究人员放弃了

对这一领域的研究，政府、企业也削减了相应的投资。

虽然如此，还是有一些具有献身精神的科学家在坚持进行相应的研究。在 20 世纪 70 年代和 80 年代早期，他们的研究结果很难得到发表，而且是散布于各种杂志之中，使得不少有意义的成果即使在发表之后，也难以被同行看到，这导致了反思期的延长。著名的 BP 算法的研究就是一个例子。

在这一段的反思中，人们发现，有一类问题是单级感知器无法解决的，这类问题是线性不可分的。要想突破线性不可分问题，必须采用功能更强的多级网络。逐渐地，一系列的基本网络模型被建立起来，形成了人工神经网络的理论基础。Minsky 的估计被证明是过分悲观的。

可以认为，这一时期一直延续到 1982 年 J. Hopfield 将 Lyapunov 函数引入人工神经网络，作为网络性能判定的能量函数为止。在这个期间，取得的主要积极成果有 Arbib 的竞争模型、Kohonen 的自组织映射、Grossberg 的自适应共振模型(ART)、Fukushima 的新认知机、Rumellhart 等人的并行分布处理模型(PDP)。

1.3.4 第二高潮期

人工神经网络研究的第二次高潮到来的标志是美国加州理工学院生物物理学家 J. Hopfield 的两篇重要论文分别于 1982 年和 1984 年在美国科学院院刊上发表。总结起来，这个期间的代表作有：

(1) 1982 年，J. Hopfield 提出循环网络，并将 Lyapunov 函数引入人工神经网络，作为网络性能判定的能量函数，阐明了人工神经网络与动力学的关系，用非线性动力学的方法来研究人工神经网络的特性，建立了人工神经网络稳定性的判别依据，指出信息被存放在网络中神经元的联接上。实际上，这里所指的信息是长期存储的信息(Long Term Memory)。这是一个突破性的进展。

(2) 1984 年，J. Hopfield 设计研制了后来被人们称为 Hopfield 网的电路。在这里，人工神经元被用放大器来实现，而联接则是用其他电子线路实现的。作为该研究的一项应用验证，它较好地解决了著名的 TSP 问题，找到了最佳解的近似解，引起了较大的轰动。

(3) 1985 年，美国加州大学圣地亚哥分校(UCSD)的 Hinton、Sejnowsky、Rumelhart 等人所在的并行分布处理(PDP)小组的研究者在 Hopfield 网络中引入了随机机制，提出所谓的 Boltzmann 机。在这里，他们借助于统计物理学的方法，首次提出了多层网的学习算法。但由于它的不确定性，其收敛速度成了较大的问题，目前主要用来使网络逃离训练中的局部极小点。

(4) 1986 年，并行分布处理小组的 Rumelhart 等研究者重新独立地提出多层网络的学习算法——BP 算法，较好地解决了多层网络的学习问题。我们之所以这样讲，是因为后来人们依次发现，类似的算法分别被 Paker 和 Werbos 在 1982 年和 1974 年独立地提出过，只不过当时没能被更多的人发现并受到应有的重视。BP 算法的提出，对人工神经网络的研究与应用起到了重大的推动作用。

这个期间，人们对神经网络的研究达到了第二次高潮，仅从 1987 年 6 月在美国加州举行的第一届神经网络国际会议就有 1 000 余名学者参加就可以看到这一点。我国在这方面的研究要

滞后一点,国内首届神经网络大会是1990年12月在北京举行的。

1.3.5 再认识与应用研究期

实际上,步入20世纪90年代后,人们发现,关于人工神经网络还有许多待解决的问题,其中包括许多理论问题。所以,近期要想用人工神经网络的方法在人工智能的研究中取得突破性的进展还为时过早。因此又开始了新一轮的再认识。

与此同时,许多研究者致力于根据实际系统的需要,改进现有的模型和基本算法,以获取较好的性能。

以物理符号系统为基础的传统的人工智能技术,模拟的是人的逻辑思维过程,人工神经网络则是对人的形象思维的模拟。按照这一说法,这两种不同的人工智能技术应该大体上拥有同样宽的应用面。但是,就目前看来,人工神经网络的应用还远不能和传统的计算并驾齐驱,它还在等待着基础研究的重大突破。

人工神经网络的不精确推理,使得它因为结果的精度较低而远远不能满足用户的需要。这在一定的程度上也影响了它的应用面。为了解决这个问题,充分发挥两种技术各自的优势,一部分研究者在系统中将其作为初步的"筛选工具",取得结果后,再用传统的方法进行求精。大量的实验表明,这种方法是一个有效的方法。

另外,目前还无法对人工神经网络的工作机理进行严格的解释,这使得它的可信度成为一个不大不小的问题。目前,大多数的研究主要集中在以下三个方面:

(1) 开发现有模型的应用,并在应用中根据实际运行情况对模型、算法加以改造,以提高网络的训练速度和运行的准确度。

(2) 希望在理论上寻找新的突破,建立新的专用/通用模型和算法。

(3) 进一步对生物神经系统进行研究,不断地丰富对人脑的认识。

练 习 题

1. 叙述人类智能的含义。

2. 什么是人工智能?研究人工智能的目的是什么?

3. 表示智能的物理符号系统的特征是什么?抽象和形式化在传统的人工智能技术中是如何发挥作用的?

4. 人工神经网络从哪几个关键方面试图去模拟人的智能?

5. 物理符号系统和联接主义的观点在探讨对人的智能的理解和表达上的根本区别是什么?

6. 试用图1-2对传统的人工智能技术和人工神经网络方法进行比较。通过此比较,你能认为人工神经网络方法将会比传统的人工智能技术有更为广阔的发展前景么?为什么?

7. 在人工神经网络的概念中,有几点是最为关键的,它们表达出了人工神经网络最本质的特征,你认为是哪几点?

8. 请总结出人工神经网络的性能特点。

9. 如何理解人工神经网络的学习能力、普化能力?

10. 通过了解人工神经网络发展的历史,你对一些基础研究工作有什么看法:是非常重要的工作,但它们仅

仅是理论工作者的责任？仅仅是少数致力于创新、发明、发现的人的工作？每个从事科学研究的人都应该给予一定的重视？

11. M. L. Minsky 和 S. Papert 从理论上证明了当时的单级感知器无法解决包括“异或”运算在内的许多简单问题，从而沉重地打击了人们对人工神经网络研究的积极性。你认为这一成果的取得和《Perceptron》一书的发表对人工神经网络研究的发展起到了什么样的历史作用？请给出你自己的评价。

12. 人脑是由许多神经元并通过它们之间的广泛联接构成的。按照图 1－2 所指出的人工神经网络模拟思维的生理过程的观点，请你考虑一下，人工神经网络应该是如何构成的？

第二章　人工神经网络基础

人工神经网络是根据人们对生物神经网络的研究成果设计出来的，它由一系列的神经元及其相应的联接构成，具有良好的数学描述，不仅可以用适当的电子线路来实现，更可以方便地用计算机程序加以模拟。本章将介绍人工神经网络的基本知识，主要包括：基本的生物神经网络模型，人工神经元模型及其典型的激活函数；人工神经网络的基本拓扑特性，存储类型（CAM－LTM，AM－STM）及映象，有导师（Supervised）训练与无导师（Unsupervised）训练等基本概念。

2.1　生物神经网络

由于人工神经网络是受生物神经网络的启发构造而成的，所以在开始讨论人工神经网络之前，有必要首先考虑人脑皮层神经系统的组成。

科学研究发现，人的大脑中大约含有 10^{11} 个生物神经元，它们通过 10^{15} 个联接被联成一个系统。每个神经元具有独立的接受、处理和传递电化学（Electrochemical）信号的能力。这种传递经由构成大脑通信系统的神经通路所完成。图 2－1 所示是生物神经元及其相互联接的典型结构。为清楚起见，在这里只画出了两个神经元，其他神经元及其相互之间的联接与此类似。

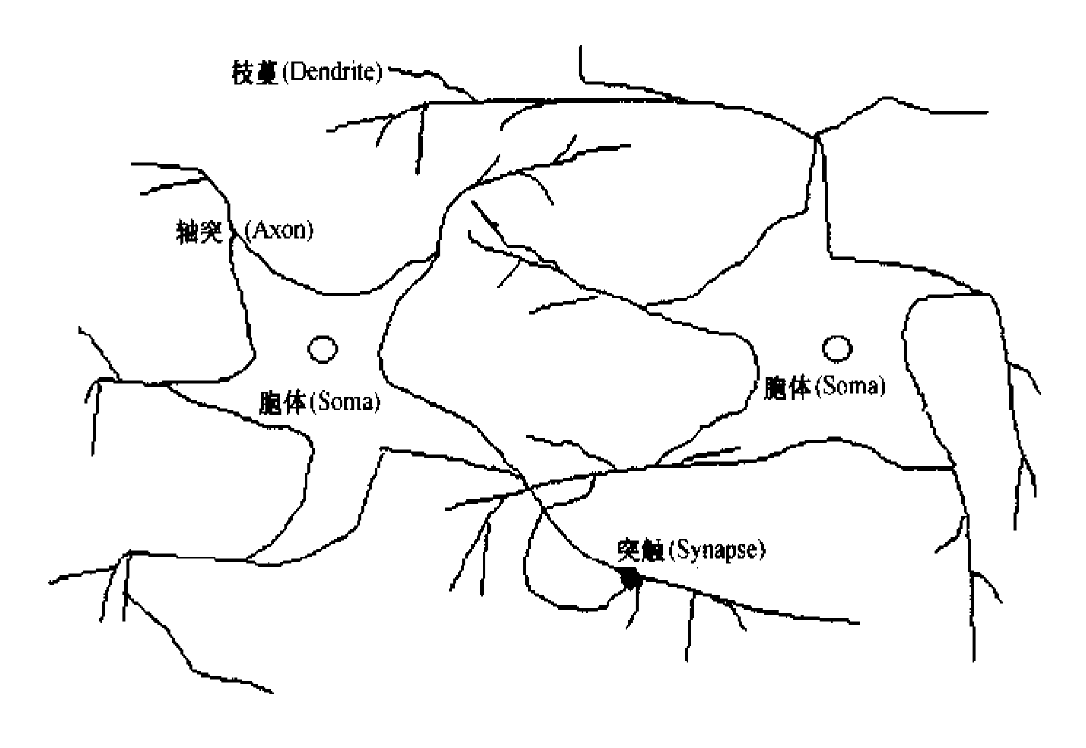


图 2－1　典型的生物神经元

如图所示，枝蔓(Dendrite)从胞体(Soma 或 Cell body)伸向其他神经元，这些神经元在被称为突触(Synapse)的联接点接受信号。在突触的接受侧，信号被送入胞体，这些信号在胞体里被综合。其中有的输入信号起刺激(Excite)作用，有的起抑制作用(Inhibit)。当胞体中接受的累加刺激超过一个阈值时，胞体就被激发，此时它沿轴突通过枝蔓向其他神经元发出信号。

在这个系统中，每一个神经元都通过突触与系统中很多其他的神经元相联系。研究认为，同一个神经元通过由其伸出的枝蔓发出的信号是相同的，而这个信号可能对接受它的不同神经元有不同的效果，这一效果主要由相应的突触决定：突触的"联接强度"越大，接受的信号就越强，反之，突触的"联接强度"越小，接受的信号就越弱。突触的"联接强度"可以随着系统受到的训练而被改变。

总结起来，生物神经系统有如下六个基本特征：

(1) 神经元及其联接；

(2) 神经元之间的联接强度决定信号传递的强弱；

(3) 神经元之间的联接强度是可以随训练而改变的；

(4) 信号可以是起刺激作用的，也可以是起抑制作用的；

(5) 一个神经元接受的信号的累积效果决定该神经元的状态；

(6) 每个神经元可以有一个"阈值"。

2.2 人工神经元

从上述可知，神经元是构成神经网络的最基本单元(构件)。因此，要想构造一个人工神经网络系统，首要任务是构造人工神经元模型。而且我们希望，这个模型不仅是简单容易实现的数学模型，而且它还应该具有上节介绍的生物神经元的六个基本特性。

2.2.1 人工神经元的基本构成

根据上述对生物神经元的讨论，我们希望人工神经元可以模拟生物神经元的一阶特性——输入信号的加权和。

对于每一个人工神经元来说，它可以接受一组来自系统中其他神经元的输入信号，每个输入对应一个权，所有输入的加权和决定该神经元的激活(Activation)状态。这里，每个权就相当于突触的"联接强度"。基本模型见图 2-2。

设 n 个输入分别用 $x_1, x_2, \cdots, x_n$ 表示，它们对应的联接权值依次为 $w_1, w_2, \cdots, w_n$，所有的输入及对应的联接权值分别构成输入向量 $\boldsymbol{X}$ 和联接权向量 $\boldsymbol{W}$：

$$\boldsymbol{X} = (x_1, x_2, \cdots, x_n)$$

$$\boldsymbol{W} = (w_1, w_2, \cdots, w_n)^T$$

用 net 表示该神经元所获得的输入信号的累积效果，为简便起见，称之为该神经元的网络输入：

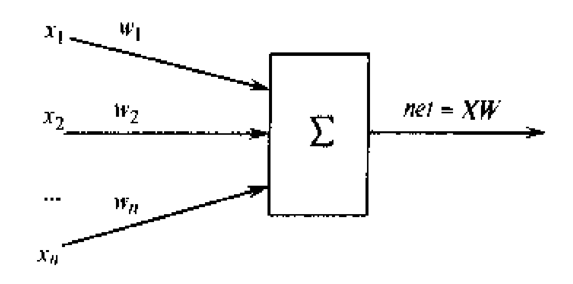


图 2-2　不带激活函数的人工神经元

$$net = \sum x_i w_i \tag{2-1}$$

写成向量形式，则有

$$net = \boldsymbol{XW} \tag{2-2}$$

2.2.2　激活函数(Activation Function)

神经元在获得网络输入后，它应该给出适当的输出。按照生物神经元的特性，每个神经元有一个阈值，当该神经元所获得的输入信号的累积效果超过阈值时，它就处于激发态；否则，应该处于抑制态。为了使系统有更宽的适用面，希望人工神经元有一个更一般的变换函数，用来执行对该神经元所获得的网络输入的变换，这就是激活函数，也可以称之为激励函数、活化函数。用 f 表示：

$$o = f(net) \tag{2-3}$$

其中，o 是该神经元的输出。由此式可以看出，函数 f 同时也用来将神经元的输出进行放大处理或限制在一个适当的范围内。典型的激活函数有线性函数、非线性斜面函数、阶跃函数、s 型函数等四种。

1. 线性函数(Linear Function)

线性函数是最基本的激活函数，它起到对神经元所获得的网络输入进行适当的线性放大的作用。它的一般形式为

$$f(net) = k \times net + c \tag{2-4}$$

式中，k 为放大系数，c 为位移，它们均为常数。图 2-3(a)所示是它的图像。

2. 非线性斜面函数(Ramp Function)

线性函数非常简单，但是它的线性性极大地降低了网络的性能，它甚至使多级网络的功能退化成单级网络的功能。因此，在人工神经网络中有必要引入非线性激活函数。

非线性斜面函数是最简单的非线性函数，实际上它是一种分段线性函数。由于它简单，所以有时也被人们采用。这种函数在于把函数的值域限制在一个给定的范围 $[-\gamma, \gamma]$ 内。

$$f(net) = \begin{cases} \gamma & \text{if } net \geqslant \theta \\ k \times net & \text{if } |net| < \theta \\ -\gamma & \text{if } net \leqslant -\theta \end{cases} \tag{2-5}$$

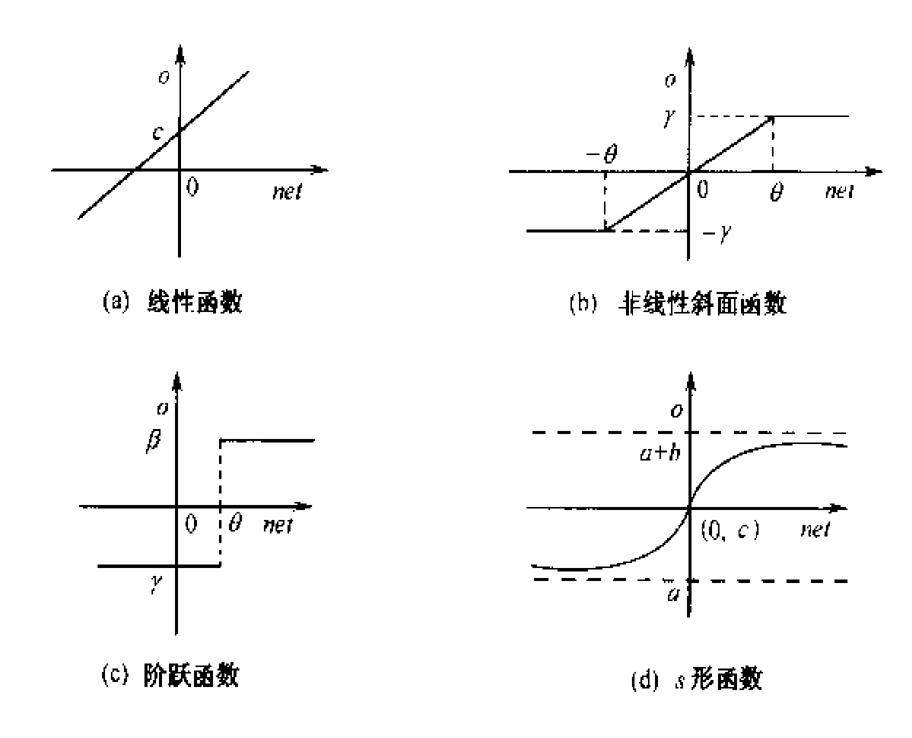


图 2-3　4 种常用的激活函数

其中,γ 为常数。一般地,规定 $\gamma>0$,它被称为饱和值,为该神经元的最大输出。图 2-3(b)所示是它的图像。

3. 阈值函数(Threshold Function)

阈值函数又叫阶跃函数,当激活函数仅用来实现判定神经元所获得的网络输入是否超过阈值 θ 时,使用此函数。

$$f(net) = \begin{cases} \beta & \text{if } net > \theta \\ -\gamma & \text{if } net \leqslant \theta \end{cases} \qquad 2-6$$

其中,β、γ、θ 均为非负实数,θ 为阈值。图 2-3(c)所示是它的图像。通常,人们用式 2-6 的二值形式:

$$f(net) = \begin{cases} 1 & \text{if } net > \theta \\ 0 & \text{if } net \leqslant \theta \end{cases} \qquad 2-7$$

有时候,还将式 2-7 中的 0 改为 -1,此时就变成了双极形式:

$$f(net) = \begin{cases} 1 & \text{if } net > \theta \\ -1 & \text{if } net \leqslant \theta \end{cases} \qquad 2-7'$$

4. s 形函数

s 形函数又叫压缩函数(Squashing Function)和逻辑斯特函数(Logistic Function),其应用最为广泛。它的一般形式为

$$f(net) = a + \frac{b}{1+\exp(-d \times net)} \qquad 2-8$$

其中,a、b、d 为常数。图 2-3(d)所示是它的图像。图中

$$c = a + \frac{b}{2}$$

它的饱和值为 a 和 $a+b$。该函数的最简单形式为

$$f(net) = \frac{1}{1 + \exp(-d \times net)}$$

此时,函数的饱和值为 0 和 1。

也可以取其他形式的函数,如双曲函数、扩充平方函数。而当取扩充平方函数

$$f(net) = \begin{cases} \dfrac{net^2}{1 + net^2} & \text{if } net > 0 \\ 0 & 其他 \end{cases}$$

时,饱和值仍然是 0 和 1。当取双曲函数

$$f(net) = \tanh(net) = \frac{e^{net} - e^{-net}}{e^{net} + e^{-net}}$$

时,饱和值则是 -1 和 1。

s 形函数之所以被广泛地应用,除了它的非线性性和处处连续可导性外,更重要的是由于该函数对信号有一个较好的增益控制:函数的值域可以由用户根据实际需要给定,在 $|net|$ 的值比较小时,$f(net)$ 有一个较大的增益;在 $|net|$ 的值比较大时,$f(net)$ 有一个较小的增益,这为防止网络进入饱和状态提供了良好的支持。

2.2.3 M-P 模型

将人工神经元的基本模型和激活函数合在一起构成人工神经元,这就是著名的 McCulloch-Pitts 模型,简称为 M-P 模型,也可以称之为处理单元(PE)。

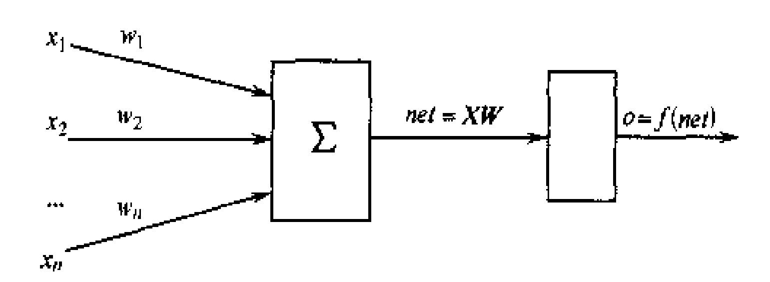


图 2-4 人工神经元

在上一章曾经提到过,UCSD 的 PDP 小组曾经将人工神经元定义得比较复杂,在本书中,为方便起见,均采用这种简化了的定义,同时简记为 AN。图 2-4 所给出的神经元在今后给出的图中均用一个结点表示。

2.3 人工神经网络的拓扑特性

为了理解方便,用结点代表神经元,用加权有向边代表从神经元到神经元之间的有向联接,

相应的权代表该联接的联接强度，用箭头代表信号的传递方向。

2.3.1 联接模式

在生物神经系统中，一个神经元接受的信号可以对其起刺激作用，也可能对其起抑制作用。在人工神经网络系统中，注意到神经元是以加权和的形式接受其他的神经元给它的信号的，所以无需特意去区分它们，只用通过赋予联接权的正、负号就可以了：

- 用正号（"+"，可省略）表示传送来的信号起刺激作用，用于增加神经元的活跃度；
- 用负号（"-"）表示传送来的信号起抑制作用，用于降低神经元的活跃度。

那么，如何组织网络中的神经元呢？研究发现，物体在人脑中的反映带有分块的特征，对一个物体，存在相应的明、暗区域。这一点启发我们可以将这些神经元分成不同的组，也就是分块进行组织。在拓扑表示中，不同的块可以被放入不同的层中。另一方面，网络应该有输入和输出，从而就有了输入层和输出层。

层次（又称为"级"）的划分，导致了神经元之间三种不同的互联模式：层（级）内联接、循环联接、层（级）间联接。

1. 层内联接

层内联接又叫做区域内（Intra-field）联接或侧联接（Lateral）。它是本层内的神经元到本层内的神经元之间的联接，可用来加强和完成层内神经元之间的竞争：当需要组内加强时，这种联接的联接权取正值；在需要实现组内竞争时，这种联接权取负值。

2. 循环联接

循环联接在这里特指神经元到自身的联接，用于不断加强自身的激活值，使本次的输出与上次的输出相关，是一种特殊的反馈信号。

3. 层间联接

层间（Inter-field）联接指不同层中的神经元之间的联接。这种联接用来实现层间的信号传递。

在复杂的网络中，层间的信号传递既可以是向前的（前馈信号），又可以是向后的（反馈信号）。一般地，前馈信号只被允许在网络中向一个方向传送；反馈信号的传送则可以自由一些，它甚至被允许在网络中循环传送。

在反馈方式中，一个输入信号通过网络变换后，产生一个输出，然后该输出又被反馈到输入端，对应于这个"新的"输入，网络又产生一个新的输出，这个输出又被再次反馈到输入端……如此重复下去。我们希望，随着这种循环的进行，在某一时刻，输入和输出不再发生变化——网络稳定了下来，那么，网络此时的输出将是网络能够给出的、最初的输入所应对应的最为理想的输出。在这个过程中，信号被一遍一遍地修复和加强，最终得到适当的结果。但是，最初的输入是一个可以"修复"的对象么？如果是，系统是否真的有能力修复它呢？这种循环是否会永远地进行下去？这就是循环网络的稳定性问题。

2.3.2 网络的分层结构

为了更好地组织网络中的神经元，我们把它们分布到各层（级）。按照上面对网络的联接的

划分，我们称侧联接引起的信号传递为横向反馈；层间的向前联接引起的信号传递为层前馈（简称前馈）；层间的向后联接引起的信号传递为层反馈。横向反馈和层反馈统称为反馈。

1．单级网

虽然单个神经元能够完成简单的模式侦测，但是为了完成较复杂的功能，还需要将大量的神经元联成网，有机的联接使它们可以协同完成规定的任务。

（1）简单单级网

最简单的人工神经网络如图 2－5 所示，该网接受输入向量 $\boldsymbol{X}$：

$$\boldsymbol{X} = (x_1, x_2, \cdots, x_n)$$

经过变换后输出向量 $\boldsymbol{Y}$：

$$\boldsymbol{O} = (o_1, o_2, \cdots, o_m)$$

图 2－5 表面上看是一个两层网，但是由于其中的输入层的神经元不对输入信号做任何处理，它们只起到对输入向量 $\boldsymbol{X}$ 的扇出作用。因此，在计算网络的层数时人们习惯上并不将它作为一层。

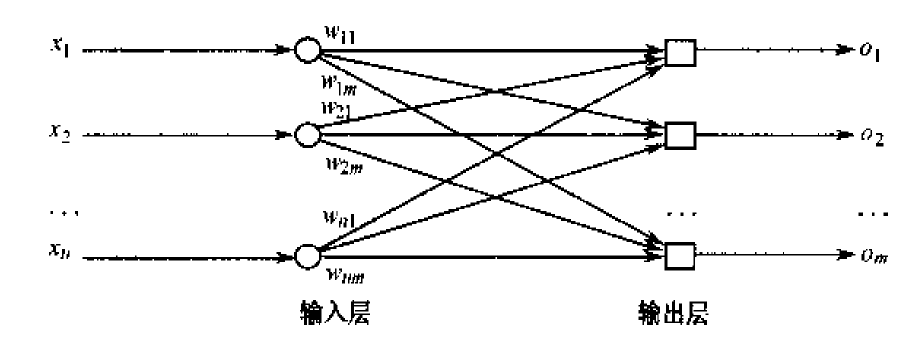


图 2－5　简单单级网

设输入层的第 i 个神经元到输出层的第 j 个神经元的联接的强度为 w_{ij}，即 $\boldsymbol{X}$ 的第 i 个分量以权重 w_{ij} 输入到输出层的第 j 个神经元中，取所有的权构成（输入）权矩阵 $\boldsymbol{W}$：

$$\boldsymbol{W} = (w_{ij})$$

输出层的第 j 个神经元的网络输入记为 net_j：

$$net_j = x_1 w_{1j} + x_2 w_{2j} + \cdots + x_n w_{nj}$$

其中，$1 \leqslant j \leqslant m$。取

$$\boldsymbol{NET} = (net_1, net_2, \cdots, net_m)$$

从而有

$$\boldsymbol{NET} = \boldsymbol{XW} \qquad 2-9$$

$$\boldsymbol{O} = \boldsymbol{F}(\boldsymbol{NET}) \qquad 2-10$$

式中 $\boldsymbol{F}$ 为输出层神经元的激活函数的向量形式。我们约定，$\boldsymbol{F}$ 对应每个神经元有一个分量，而且它的第 j 个分量对应作用在 $\boldsymbol{NET}$ 的第 j 个分量 net_j 上。一般情况下，不对其各个分量加以区分，认为它们是相同的。对此，今后不再说明。

根据信息在网络中的流向，称 $\boldsymbol{W}$ 是从输入层到输出层的联接权矩阵，而这种只有一级联接

矩阵的网络叫做简单单级网。为方便起见，有时将网络中的联接权矩阵与其到达方相关联。例如，上述的 $\boldsymbol{W}$ 就可以被称为输出层权矩阵。

(2) 单级横向反馈网

在简单单级网的基础上，在其输出层加上侧联接就构成单级横向反馈网。如图 2－6 所示。

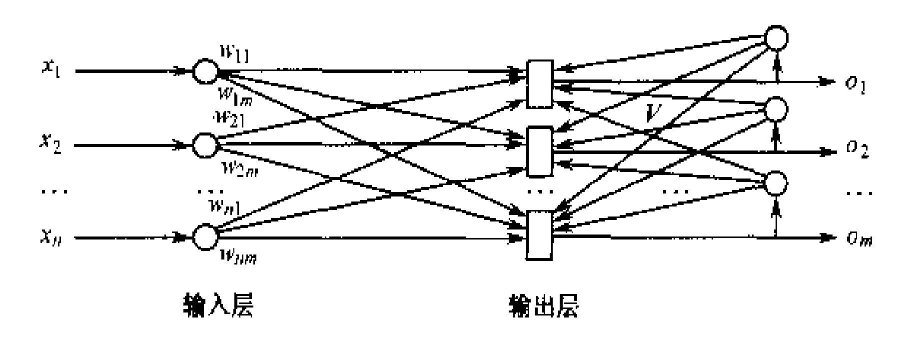


图 2－6　单级横向反馈网

设输出层的第 i 个神经元到输出层的第 j 个神经元的联接的强度为 v_{ij}，即 $\boldsymbol{O}$ 的第 i 个分量以权重 v_{ij} 输入到输出层的第 j 个神经元中。取所有的权构成侧联接权矩阵 $\boldsymbol{V}$：

$$\boldsymbol{V} = (v_{ij})$$

则

$$\boldsymbol{NET} = \boldsymbol{XW} + \boldsymbol{OV} \qquad 2-11$$

$$\boldsymbol{O} = F(\boldsymbol{NET}) \qquad 2-12$$

在此网络中，对一个输入，如果网络最终能给出一个不变的输出，也就是说，网络的运行逐渐会达到稳定，则称该网络是稳定的；否则称之为不稳定的。网络的稳定性问题是困扰有反馈信号的网络的性能的重要问题。因此，稳定性判定是一个非常重要的问题。

由于信号的反馈，使得网络的输出随时间的变化而不断变化，所以时间参数有时候也是在研究网络运行中需要特别给予关注的一个重要参数。下面假定，在网络的运行过程中有一个主时钟，网络中的神经元的状态在主时钟的控制下同步变化。在这种假定下，有

$$\boldsymbol{NET}(t+1) = \boldsymbol{X}(t)\boldsymbol{W} + \boldsymbol{O}(t)\boldsymbol{V} \qquad 2-13$$

$$\boldsymbol{O}(t+1) = F(\boldsymbol{NET}(t+1)) \qquad 2-14$$

其中，当 $t = 0$ 时 $\boldsymbol{O}(0) = \boldsymbol{0}$。

读者自己可以考虑 $\boldsymbol{X}$ 仅在 $t = 0$ 时加在网上的情况。

2. 多级网

研究表明，单级网的功能是有限的，适当地增加网络的层数是提高网络计算能力的一个途径，这也部分地模拟了人脑的某些部位的分级结构特征。

从拓扑结构上来看，多级网是由多个单级网联接而成的。

(1) 层次划分

图 2－7 所示是一个典型的多级前馈网，又叫做非循环多级网络。在这种网络中，信号只被允许从较低层流向较高层。我们约定，用层号确定层的高低：层号较小者，层次较低；层号较大

者,层次较高。各层的层号按如下方式递归定义:

• 输入层:与单级网络一样,该层只起到输入信号的扇出作用。所以在计算网络的层数时不被记入。该层负责接收来自网络外部的信息,被记作第0层。

• 第 j 层:第 $j-1$ 层的直接后继层($j>0$),它直接接受第 $j-1$ 层的输出。

• 输出层:它是网络的最后一层,具有该网络的最大层号,负责输出网络的计算结果。

• 隐藏层:除输入层和输出层以外的其他各层叫隐藏层。隐藏层不直接接受外界的信号,也不直接向外界发送信号。

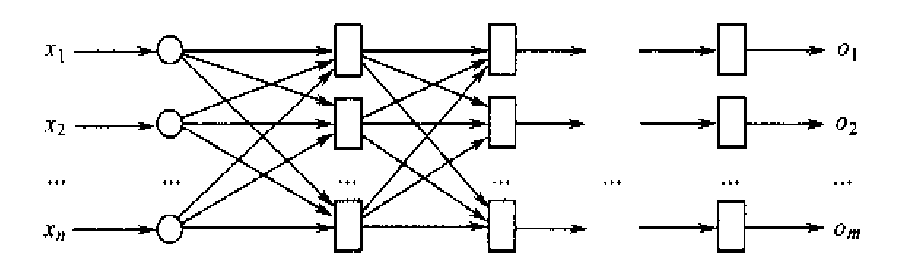


图 2-7 多级前馈网

此外,我们约定:

① 输出层的层号为该网络的层数,并称一个输出层号为 n 的网络为 n 层网络或 n 级网络。

② 第 $j-1$ 层到第 j 层的联接矩阵为第 j 层联接矩阵,输出层对应的矩阵叫输出层联接矩阵。今后,在需要的时候,一般用 $\boldsymbol{W}^{(j)}$ 表示第 j 层矩阵。

(2) 非线性激活函数

前面曾经提到过,非线性激活函数在多级网络中起着非常重要的作用。实际上,它除了能够根据需要对网络中各神经元的输出进行变换外,还使得多级网络的功能超过单级网络,为解决人工神经网络所面临的线性不可分问题提供了基础。

增加网络的层数在于提高网络的计算能力。但是,如果使用线性激活函数,则多级网的功能不会超过单级网的功能。事实上,设有一 n 层网络,$\boldsymbol{X}$ 是其输入向量,$\boldsymbol{W}^{(1)}$、$\boldsymbol{W}^{(2)}$、…、$\boldsymbol{W}^{(n)}$ 是各级联接矩阵,$\boldsymbol{NET}_1$、$\boldsymbol{NET}_2$、…、$\boldsymbol{NET}_n$ 分别是各级的网络输入向量,$\boldsymbol{F}_1$、$\boldsymbol{F}_2$、…、$\boldsymbol{F}_n$ 为各级神经元的激活函数,现假定它们均是线性的:

$$\boldsymbol{F}_i(\boldsymbol{NET}_i) = \boldsymbol{K}_i\boldsymbol{NET}_i + \boldsymbol{A}_i \qquad 1 \leqslant i \leqslant n \tag{2-15}$$

其中,$\boldsymbol{K}_i$、$\boldsymbol{A}_i$ 是常数向量,且这里的 $\boldsymbol{K}_i\boldsymbol{NET}_i$ 有特殊的意义,它表示 $\boldsymbol{K}_i$ 与 $\boldsymbol{NET}_i$ 的分量对应相乘,结果仍然是同维向量。

令

$$\boldsymbol{K}_i = (k_1, k_2, \cdots, k_{ni})$$

$$\boldsymbol{NET}_i = (net_1, net_2, \cdots, net_{ni})$$

则

$$\boldsymbol{K}_i\boldsymbol{NET}_i = (k_1\ net_1, k_2\ net_2, \cdots, k_{ni}\ net_{ni}) \tag{2-16}$$

网络的输出向量为

$$
\begin{aligned}
\boldsymbol{O} &= \boldsymbol{F}_n(\ldots \boldsymbol{F}_3(\boldsymbol{F}_2(\boldsymbol{F}_1(\boldsymbol{NET}_1)))\ldots) \\
&= \boldsymbol{F}_n(\ldots \boldsymbol{F}_3(\boldsymbol{F}_2(\boldsymbol{K}_1\boldsymbol{X}\,\boldsymbol{W}^{(1)} + \boldsymbol{A}_1))\ldots) \\
&= \boldsymbol{F}_n(\ldots \boldsymbol{F}_3(\boldsymbol{K}_2(\boldsymbol{K}_1\boldsymbol{X}\,\boldsymbol{W}^{(1)} + \boldsymbol{A}_1) + \boldsymbol{W}^{(2)} + \boldsymbol{A}_2)\ldots) \\
&= \boldsymbol{F}_n(\ldots \boldsymbol{F}_3(\boldsymbol{K}_2\boldsymbol{K}_1\boldsymbol{X}\,\boldsymbol{W}^{(1)}\boldsymbol{W}^{(2)} + \boldsymbol{K}_2\boldsymbol{A}_1\boldsymbol{W}^{(2)} + \boldsymbol{A}_2)\ldots) \\
&= \boldsymbol{F}_n(\ldots(\boldsymbol{K}_3(\boldsymbol{K}_2\boldsymbol{K}_1\boldsymbol{X}\,\boldsymbol{W}^{(1)}\boldsymbol{W}^{(2)} + \boldsymbol{K}_2\boldsymbol{A}_1\boldsymbol{W}^{(2)} + \boldsymbol{A}_2)\boldsymbol{W}^{(3)} + \boldsymbol{A}_3)\ldots) \\
&= \boldsymbol{F}_n(\ldots(\boldsymbol{K}_3\boldsymbol{K}_2\boldsymbol{K}_1\boldsymbol{X}\,\boldsymbol{W}^{(1)}\boldsymbol{W}^{(2)}\boldsymbol{W}^{(3)} + \boldsymbol{K}_3\boldsymbol{K}_2\boldsymbol{A}_1\boldsymbol{W}^{(2)} + \boldsymbol{K}_3\boldsymbol{A}_2\boldsymbol{W}^{(3)} + \boldsymbol{A}_3)\ldots) \\
&\quad \cdots\cdots \\
&= \boldsymbol{K}_n\ldots\boldsymbol{K}_3\boldsymbol{K}_2\boldsymbol{K}_1\boldsymbol{X}\,\boldsymbol{W}^{(1)}\boldsymbol{W}^{(2)}\boldsymbol{W}^{(3)}\ldots\boldsymbol{W}^{(n)} \\
&\quad + \boldsymbol{K}_n\ldots\boldsymbol{K}_3\boldsymbol{K}_2\boldsymbol{A}_1\boldsymbol{W}^{(2)}\boldsymbol{W}^{(3)}\ldots\boldsymbol{W}^{(n)} \\
&\quad + \boldsymbol{K}_n\ldots\boldsymbol{K}_3\boldsymbol{A}_2\boldsymbol{W}^{(3)}\ldots\boldsymbol{W}^{(n)} \\
&\quad \cdots\cdots \\
&\quad + \boldsymbol{K}_n\ldots\boldsymbol{K}_{i+1}\boldsymbol{A}_i\boldsymbol{W}^{(i+1)}\ldots\boldsymbol{W}^{(n)} \\
&\quad \cdots\cdots \\
&\quad + \boldsymbol{K}_n\boldsymbol{A}_{n-1}\boldsymbol{W}^{(n)} \\
&\quad + \boldsymbol{A}_n \\
&= \boldsymbol{KXW} + \boldsymbol{A}
\end{aligned}
$$

其中

$$
\begin{aligned}
\boldsymbol{K} &= \boldsymbol{K}_n\ldots\boldsymbol{K}_3\boldsymbol{K}_2\boldsymbol{K}_1 \\
\boldsymbol{W} &= \boldsymbol{W}^{(1)}\boldsymbol{W}^{(2)}\boldsymbol{W}^{(3)}\ldots\boldsymbol{W}^{(n)} \\
\boldsymbol{A} &= \boldsymbol{K}_n\ldots\boldsymbol{K}_2\boldsymbol{K}_1\boldsymbol{A}_1\boldsymbol{W}^{(2)}\boldsymbol{W}^{(3)}\ldots\boldsymbol{W}^{(n)} \\
&\quad + \boldsymbol{K}_n\ldots\boldsymbol{K}_3\boldsymbol{A}_2\boldsymbol{W}^{(3)}\ldots\boldsymbol{W}^{(n)} \\
&\quad \cdots\cdots \\
&\quad + \boldsymbol{K}_n\ldots\boldsymbol{K}_{i+1}\boldsymbol{A}_i\boldsymbol{W}^{(i+1)}\ldots\boldsymbol{W}^{(n)} \\
&\quad \cdots\cdots \\
&\quad + \boldsymbol{K}_n\boldsymbol{A}_{n-1}\boldsymbol{W}^{(n)} \\
&\quad + \boldsymbol{A}_n
\end{aligned}
$$

上述式子中，向量 $\boldsymbol{K}_i$ 之间的运算遵循式 2－16 的约定。

从上述推导可见，这个多级网相当于一个激活函数为 $F(\boldsymbol{NET}) = \boldsymbol{KNET} + \boldsymbol{A} = \boldsymbol{KXW} + \boldsymbol{A}$，联接矩阵为 $\boldsymbol{W}$ 的简单单级网络。显然，如果网络使用的是非线性激活函数，则不会出现上述问题。因此说，非线性激活函数是多级网络的功能超过单级网络的保证。

3. 循环网

如果将输出信号反馈到输入端，就可构成一个多层的循环网络，如图 2－8 所示。其中的反馈联接还可以是其他的形式。

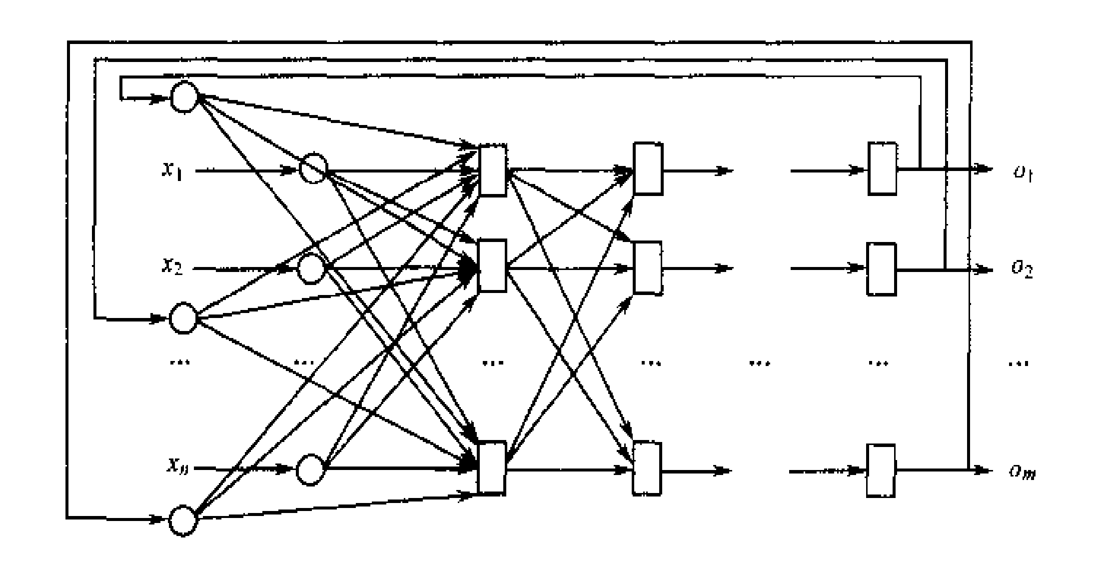


图 2-8　多级循环网

实际上,引入反馈的主要目的是解决非循环网络对上一次的输出无记忆的问题。在非循环网络中,输出仅仅由当前的输入和权矩阵决定,而和较前的计算无关。在循环网中,它需要将输出送回到输入端,从而使当前的输出受到上次输出的影响,进而又受到前一个输入的影响,如此形成一个迭代。也就是说,在这个迭代过程中,输入的原始信号被逐步地"加强"、被"修复"。

这种性能,在一定的程度上反映了人的大脑的短期记忆特征——看到的东西不是一下子就从脑海里消失的。

当然,前面曾经提到过,这种反馈信号会引起网络输出的不断变化。如果这种变化逐渐减小,并且最后能消失,一般来说,这种变化就是我们所希望的变化。当变化最后消失时,我们称网络达到了平衡状态。如果这种变化不能消失,则称该网络是不稳定的。

2.4　存储与映射

人工神经网络是用来处理信息的。可以认为,所有的信息都是以模式的形式出现的:输入向量是模式,输出向量是模式,同层的神经元在某一时刻的状态是模式,所有的神经元在某一时刻的状态是模式,网络中任意层的权矩阵、权矩阵所含的向量都是模式。在循环网络中,所有的神经元的状态沿时间轴展开,这就形成一个模式系列。所以说,在人工神经网络中,有两种类型的模式:空间模式(Spatial Model)和时空模式(Spatialtemporal Model)。网络所有的神经元在某一时刻的状态所确定的网络在该时刻的状态叫做空间模式;以时间维为轴展开的空间模式系列叫做时空模式,这两种模式之间的关系如同一个画面与整个影片的关系。仅在考虑循环网络的稳定性和网络训练的收敛过程时涉及到时空模式,一般情况下,只研究空间模式。

在日常生活中,当寻找一单位时,需要知道它的地址,然后根据地址去访问它;在计算机系统中,目前习惯的也是通过地址去存放和取出数据。实际上,在人工神经网络技术中,空间模式的存取还有另外两种方式。所以,按照信息的存放与提取的方式的不同,空间模式共有三种存储类

型。

1. RAM方式

RAM方式即随机访问方式(Random Access Memory)。这种方式就是现有的计算机中的数据访问方式。这种方式需要按地址去存取数据,即将地址映射到数据。

2. CAM方式

CAM方式即内容寻址方式(Content Addressable Memory)。在这种方式下,数据自动地找到它的存放位置。换句话说,就是将数据变换成它应存放的位置,并执行相应的存储。例如,在后面介绍的人工神经网络的训练算法中,样本数据被输入后,它的内容被自动存储起来,虽然现在还不知道它们具体是如何被存放的。这种方式是将数据映射到地址。

3. AM方式

AM方式即相联存储方式(Associative Memory)。这种方式是数据到数据的直接转换。在人工神经网络的正常工作阶段,输入模式(向量)经过网络的处理,被转换成输出模式(向量)。这种方式是将数据映射到数据。

后面的两种方式是人工神经网络的工作方式。在学习/训练期间,人工神经网络以CAM方式工作:它将样本数据以各层神经元之间的连接权矩阵的稳定状态存放起来。由于权矩阵在大多数网络的正常运行阶段是一直被保存不变的,所以权矩阵又被称为网络的长期存储(Long Term Memory,简记为LTM)。

网络在正常工作阶段是以AM方式工作的。此时,输入模式被转换成输出模式。由于输出模式是以网络输出层的神经元的状态表示出来的,而在下一个时刻,或者在下一个新的输入向量加到网络上的时候,这一状态将被改变,所以,称由神经元的状态表示的模式为短期存储(Short Term Memory,简记为STM)。

输入向量与输出向量的对应关系是网络的设计者所关心的另一个问题。和模式完善相对应,人工神经网络可以实现还原型映射。如果此时训练网络的样本集为向量集合:

$$\{\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_n\} \tag{2-17}$$

在理想情况下,该网络在完成训练后,其权矩阵存放的将是上式所给的向量集合。此时网络实现的映射将是自相联(Auto-associative)映射。

人工神经网络还可以实现变换型和分类型映射。如果此时训练网络的样本集为向量对组成的集合:

$$\{(\boldsymbol{A}_1, \boldsymbol{B}_1), (\boldsymbol{A}_2, \boldsymbol{B}_2), \cdots, (\boldsymbol{A}_n, \boldsymbol{B}_n)\} \tag{2-18}$$

则在理想情况下,该网络在完成训练后,其权矩阵存放的将是上式所给的向量集合所蕴含的对应关系,也就是输入向量 $\boldsymbol{A}_i$ 与输出向量 $\boldsymbol{B}_i$ 的映射关系。此时网络实现的映射是异相联(Hetero-associative)映射。

由样本集确定的映射关系被存放在网络中后,当一个实际的输入向量被输入时,网络应能完成相应的变换。对异相联映射来说,如果网络中存放的集合为2-18,理想情况下,当输入向量为 $\boldsymbol{A}_i$ 时,网络应该输出向量 $\boldsymbol{B}_i$。实际上,在许多时候,网络输出的并不是 $\boldsymbol{B}_i$,而是 $\boldsymbol{B}_i$ 的一个近似向量,这是人工神经网络计算的不精确性造成的。

当输入向量 $\boldsymbol{A}$ 不是集合式 2－18 的某个元素的第 1 分量时，网络会根据集合式 2－18 给出 $\boldsymbol{A}$ 对应的理想输出的近似向量 $\boldsymbol{B}$。多数情况下，如果在集合式 2－18 中不存在这样的元素（$\boldsymbol{A}_k$，$\boldsymbol{B}_k$），使得

$$\boldsymbol{A}_i \leqslant \boldsymbol{A}_k \leqslant \boldsymbol{A}$$

或者

$$\boldsymbol{A} \leqslant \boldsymbol{A}_k \leqslant \boldsymbol{A}_j$$

且

$$\boldsymbol{A}_i \leqslant \boldsymbol{A} \leqslant \boldsymbol{A}_j$$

则向量 $\boldsymbol{B}$ 是 $\boldsymbol{B}_i$ 与 $\boldsymbol{B}_j$ 的插值。

2.5 人工神经网络的训练

人工神经网络最具有吸引力的特点是它的学习能力。1962 年，Rosenblatt 给出了人工神经网络著名的学习定理：人工神经网络可以学会它可以表达的任何东西。但是，人工神经网络的表达能力是有限的，这就大大地限制了它的学习能力。

人工神经网络的学习过程就是对它的训练过程。所谓训练，就是在将由样本向量构成的样本集合（被简称为样本集、训练集）输入到人工神经网络的过程中，按照一定的方式去调整神经元之间的联接权，使得网络能将样本集的内涵以联接权矩阵的方式存储起来，从而使得在网络接受输入时，可以给出适当的输出。

从学习的高级形式来看，一种是有导师学习，另一种是无导师学习，而前者看起来更为普遍些。无论是学生到学校接受老师的教育，还是自己读书学习，都属于有导师学习。还有不少时候，人们是经过一些实际经验不断总结学习的，也许这些应该算做无导师学习。

从学习的低级形式来看，恐怕只有无导师的学习形式。因为到目前为止，我们还没能发现在生物神经系统中有导师学习是如何发生的。在那里还找不到“导师”的存在并发挥作用的迹象，所有的只是自组织、自适应的运行过程。

2.5.1 无导师学习

无导师学习（Unsupervised Learning）与无导师训练（Unsupervised Training）相对应。该方法最早由 Kohonen 等人提出。

虽然从学习的高级形式来看，人们熟悉和习惯的是有导师学习，但是，人工神经网络模拟的是人脑思维的生物过程。而按照上述说法，这个过程应该是无导师学习的过程。所以，无导师的训练方式是人工神经网络的较具说服力的训练方法。

无导师训练方法不需要目标，其训练集中只含一些输入向量，训练算法致力于修改权矩阵，以使网络对一个输入能够给出相容的输出，即相似的输入向量可以得到相似的输出向量。

在训练过程中，相应的无导师训练算法用来将训练的样本集合中蕴含的统计特性抽取出来，并以神经元之间的联接权的形式存于网络中，以使网络可以按照向量的相似性进行分类。

虽然用一定的方法对网络进行训练后，可收到较好的效果。但是，对给定的输入向量来说，它们应被分成多少类，某一个向量应该属于哪一类，这一类的输出向量的形式是什么样的，等等，都是难以事先给出的。从而在实际应用中，还要求进行将其输出变换成一个可理解的形式的工作。另外，其运行结果的难以预测性也给此方法的使用带来了一定的障碍。

主要的无导师训练方法有 Hebb 学习律、竞争与协同(Competitive and Cooperative)学习、随机联接学习(Randomly Connected Learning)等。其中 Hebb 学习律是最早被提出的学习算法，目前的大多数算法都来源于此算法。

Hebb 算法是 D. O. Hebb 在 1961 年提出的，该算法认为，联接两个神经元的突触的强度按下列规则变化：

当两个神经元同时处于激发状态时被加强，否则被减弱。可用如下数学表达式表示：

$$\boldsymbol{W}_{ij}(t+1) = \boldsymbol{W}_{ij}(t) + \alpha o_i(t) o_j(t) \qquad 2-19$$

其中，$\boldsymbol{W}_{ij}(t+1)$、$\boldsymbol{W}_{ij}(t)$分别表示神经元 AN_i 到 AN_j 的联接在时刻 $t+1$ 和时刻 t 的强度，$o_i(t)$、$o_j(t)$为这两个神经元在时刻 t 的输出，α 为给定的学习率。

2.5.2 有导师学习

在人工神经网络中，除了上面介绍的无导师训练外，还有有导师训练。有导师学习(Supervised Learning)与有导师训练(Supervised Training)相对应。

虽然有导师训练从生物神经系统的工作原理来说，因难以解释而受到一定的非议，但是，目前看来，有导师学习却是非常成功的。因此，需要对有导师学习方法进行研究。

在这种训练中，要求用户在给出输入向量的同时，还必须同时给出对应的理想输出向量。所以，采用这种训练方式训练的网络实现的是异相联的映射。输入向量与其对应的输出向量构成一个“训练对”。

有导师学习的训练算法的主要步骤包括：

(1) 从样本集合中取一个样本($\boldsymbol{A}_i$，$\boldsymbol{B}_i$)；

(2) 计算出网络的实际输出 $\boldsymbol{O}$；

(3) 求 $\boldsymbol{D} = \boldsymbol{B}_i - \boldsymbol{O}$；

(4) 根据 $\boldsymbol{D}$ 调整权矩阵 $\boldsymbol{W}$；

(5) 对每个样本重复上述过程，直到对整个样本集来说，误差不超过规定范围。

有导师训练算法中，最为重要、应用最普遍的是 Delta 规则。1960 年，Widrow 和 Hoff 就给出了如下形式的 Delta 规则：

$$\boldsymbol{W}_{ij}(t+1) = \boldsymbol{W}_{ij}(t) + \alpha(y_j - a_j(t))o_i(t) \qquad 2-20$$

也可以写成

$$\boldsymbol{W}_{ij}(t+1) = \boldsymbol{W}_{ij}(t) + \Delta\boldsymbol{W}_{ij}(t)$$

$$\Delta\boldsymbol{W}_{ij}(t) = \alpha\delta_j o_i(t)$$

$$\delta_j = y_j - o_j(t)$$

Grossberg 的写法为

$$\Delta W_{ij}(t) = \alpha a_i(t)(o_j(t) - W_{ij}(t))$$

更一般的 Delta 规则为

$$\Delta W_{ij}(t) = g(a_i(t), y_j, o_i(t), W_{ij}(t)) \qquad 2-21$$

上述式子中，$W_{ij}(t+1)$、$W_{ij}(t)$分别表示神经元 AN_i 到 AN_j 的联接在时刻 $t+1$ 和时刻 t 的强度，$o_i(t)$、$o_j(t)$ 为这两个神经元在时刻 t 的输出，y_j 为神经元AN_j 的理想输出，$a_i(t)$、$a_j(t)$ 分别为神经元 AN_i 和 AN_j 的激活状态，α 为给定的学习率。

练　习　题

1. 叙述生物神经系统的 6 个基本特征。

2. M－P 模型是如何模拟生物神经元的一阶特性的？其中的激活函数有什么作用？

3. s 形函数有什么特征？它的突出优点是什么？除了本章给出的 s 形函数外，你还知道哪些 s 形函数？请证明式 2－8 的饱和值为 a 和 $a+b$，且 $c = a + b/2$。

4. 人工神经网络中有哪几种联接？它们各起什么作用？

5. 试分析人工神经网络分层结构的作用。

6. 长期存储(LTM)与短期存储(STM)各表示什么？它们是如何在人工神经网络中存在并表现出来的？

7. 请举出自相联映射和异相联映射的实例来。

8. Rosenblatt 给出了著名的人工神经网络学习定理，分析并详细说明该定理表达出来的意思。

9. 空间模式的存取有三种不同的方式，请问它们各有什么特点？你能在哪里找到它们的应用？

10. 无导师学习的特征是什么？网络是怎样在无导师的条件下完成学习的？

11. 当前的研究表明，从微观结构看，人脑中进行的学习是无导师的。所以，人们在人工神经网络中引入了无导师学习，在实际生活中，也存在着无导师学习的情况，请举出人类生活中无导师学习的例子。

12. 有导师学习的特征是什么？网络是怎样在有导师的条件下完成学习的？

13. 是否可以在一个系统中既实现无导师学习又实现有导师学习？如果可以，你打算如何实现？如果不可以，请说明原因。

14. 人们已经提出了一系列的人工神经网络模型，请根据本章的内容，预测一下，各种模型将通过什么进行区分？

15. 如果请你用计算机程序去完成对一个人工神经网络的模拟实现，你将如何去表示网络中的神经元及它们之间的联接？

第三章 感知器

感知器(Perceptron)是最早被设计并被实现的人工神经网络。

作为对人工神经网络的初步认识,本章介绍感知器与人工神经网络的早期发展;线性可分问题与线性不可分问题;Hebb 学习律,Delta 规则,感知器的训练算法。

3.1 感知器与人工神经网络的早期发展

1943 年,McCulloch 和 Pitts 发表了他们关于人工神经网络的第一个系统研究。1947 年,他们又开发出一个用于模式识别的网络模型——感知器,通常就叫做 M-P 模型,即阈值加权和模型。图 3-1 所示是一个单输出的感知器,不难看出,它实质上就是一个典型的人工神经元。按照 M-P 模型的要求,该人工神经元的激活函数是阶跃函数。为了适应更广泛的问题的求解,可以按如图 3-2 所示的结构,用多个这样的神经元构成一个多输出的感知器。

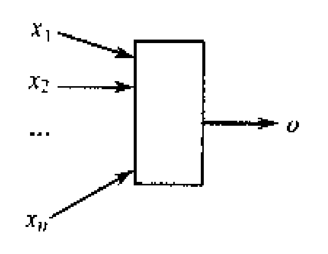


图 3-1 单输出的感知器(M-P 模型)

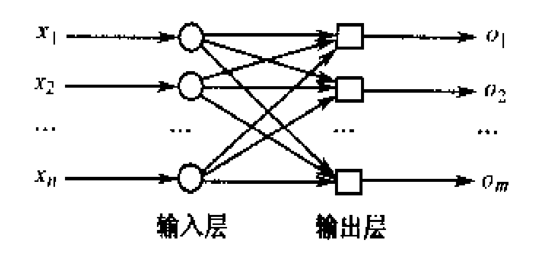


图 3-2 输出感知器

由于感知器的出现,使得人工神经网络在 20 世纪 40 年代初步地展现出它的功能及其诱人的发展前景。M-P 模型的建立,标志着已经有了构造人工神经网络系统的最基本构件。人工神经网络研究的这一初步成功,使得人们开始致力于探索如何用硬件和软件去实现神经生理学家所发现的神经网络模型。到了 20 世纪 60 年代,感知器的研究获得了较大的发展,并展示了较为乐观的前景。1962 年,Rosenblatt 证明了关于 Perceptron 的学习能力的重要结论。他向人们宣布:人工神经网络可以学会它能表示的任何东西。正当人们为取得的巨大进展而高兴的时候,却发现了有许多问题用人工神经网络是无法解决的。Minsky 严格地对问题进行了分析,证明了单级网无法解决“异或”等最基本的问题。这使得人工神经网络一下子被打入了冷宫,使其发展从第一个高潮期进入了反思期。有人认为,Minsky 的悲观观点就象是在人工神经网络研究的历史长河中筑起了一道大坝,“研究”有机会在此积蓄力量,为今后的发展打下必要的基础。实际上,人工神经网络的发展史,也表现出人们对问题的“认识、实践、再认识、再实践”的过程。

3.2 感知器的学习算法

感知器的学习是有导师学习。

感知器的训练算法的基本原理来源于著名的 Hebb 学习律,其基本思想是:逐步地将样本集中的样本输入到网络中,根据输出结果和理想输出之间的差别来调整网络中的权矩阵。作为本书介绍的第一个人工神经网络模型,本节将按离散单输出感知器、离散多输出感知器、连续多输出感知器分别详细地叙述相应的算法,目的在于引导读者了解和掌握用计算机程序实现人工神经网络的基本方法。

3.2.1 离散单输出感知器训练算法

参考图 3-1,设 F 为相应的激活函数,先按阈值函数来考虑有关问题。约定今后对这类自变量及其函数的值、向量分量的值只取 0 和 1 的函数和向量,都简称为二值的。按照这种约定,使用阈值函数作为激活函数的网络就是二值网络。另外,设 $\boldsymbol{W}$ 为网络的权向量,$\boldsymbol{X}$ 为输入向量:

$$\boldsymbol{W} = (w_1, w_2, \cdots, w_n)$$

$$\boldsymbol{X} = (x_1, x_2, \cdots, x_n)$$

网络的训练样本集为

$$\{(\boldsymbol{X}, \boldsymbol{Y}) \mid \boldsymbol{X} \text{ 为输入向量}, \boldsymbol{Y} \text{ 为 } \boldsymbol{X} \text{ 对应的输出}\}$$

此时,可有如下离散单输出感知器训练算法:

算法 3-1 离散单输出感知器训练算法

1 初始化权向量 $\boldsymbol{W}$;

2 重复下列过程,直到训练完成:

  2.1 对样本集中的每一个样本$(\boldsymbol{X}, \boldsymbol{Y})$,重复如下过程:

    2.1.1 输入 $\boldsymbol{X}$;

    2.1.2 计算 $\boldsymbol{O} = \boldsymbol{F}(\boldsymbol{XW})$;

    2.1.3 如果输出不正确,则

      当 $o=0$ 时,取 $\boldsymbol{W} = \boldsymbol{W} + \boldsymbol{X}$;

      当 $o=1$ 时,取 $\boldsymbol{W} = \boldsymbol{W} - \boldsymbol{X}$

上述算法中,当 $o = 0$ 时,按 $\boldsymbol{W} + \boldsymbol{X}$ 修改权向量 $\boldsymbol{W}$。这是因为,理想输出本来应该是 1,但现在却是 0,所以,相应的权应该增加,而且是增加对该样本的实际输出真正有贡献的权。当 $o = 1$ 时恰好相反,详细情况读者自己可以进行分析。

3.2.2 离散多输出感知器训练算法

参考图 3-2,设 F 为网络中神经元的激活函数,$\boldsymbol{W}$ 为权矩阵,w_{ij} 为输入向量的第 i 个分量到第 j 个神经元的联接权:

$$\boldsymbol{W} = \{w_{ij}\}$$

网络的训练样本集为:

$$\{(\boldsymbol{X},\boldsymbol{Y}) \mid \boldsymbol{X} \text{ 为输入向量}, \boldsymbol{Y} \text{ 为 } \boldsymbol{X} \text{ 对应的输出}\}$$

这里,假定 $\boldsymbol{X}$、$\boldsymbol{Y}$ 分别是维数为 n 的输入向量和维数为 m 的理想输出向量:

$$\boldsymbol{X} = (x_1, x_2, \cdots, x_n)$$

$$\boldsymbol{Y} = (y_1, y_2, \cdots, y_m)$$

之所以称 $\boldsymbol{Y}$ 为输入向量 $\boldsymbol{X}$ 的理想输出向量,是为了与网络的实际输出向量 $\boldsymbol{O}$ 相区别。之所以将($\boldsymbol{X}$, $\boldsymbol{Y}$)选做样本,是因为在实际系统(这里指人工神经网络所模拟的对象)运行中,当遇到输入向量 $\boldsymbol{X}$ 时,系统会输出向量 $\boldsymbol{Y}$。显然,也希望相应的人工神经网络在接受到输入向量 $\boldsymbol{X}$ 时,也能输出向量 $\boldsymbol{Y}$。但是,由于人工神经网络是对实际系统的模拟,再加上它的不精确性,它实际上很难在接受输入向量 $\boldsymbol{X}$ 时,精确地输出向量 $\boldsymbol{Y}$。此时网络会输出 $\boldsymbol{Y}$ 的一个近似向量 $\boldsymbol{O}$:

$$\boldsymbol{O} = (o_1, o_2, \cdots, o_m)$$

为了区分向量 $\boldsymbol{Y}$ 和 $\boldsymbol{O}$,称 $\boldsymbol{Y}$ 是 $\boldsymbol{X}$ 对应的理想输出向量;$\boldsymbol{O}$ 是 $\boldsymbol{X}$ 对应的实际输出向量。在离散多输出感知器中,由于它含有多个输出神经元,因此,在训练算法的组织上不能再沿用算法 3-1 的实现方式。但是,仍然遵循相同的原理,按照相同的思想去实施对各联接权的调整。具体算法如下:

算法 3-2　离散多输出感知器训练算法

1　初始化权矩阵 $\boldsymbol{W}$;

2　重复下列过程,直到训练完成:

　2.1　对样本集中的每一个样本($\boldsymbol{X}$, $\boldsymbol{Y}$),重复如下过程:

　　2.1.1　输入 $\boldsymbol{X}$;

　　2.1.2　计算 $\boldsymbol{O} = F(\boldsymbol{XW})$;

　　2.1.3　for $i = 1$ to m 执行如下操作:

　　　if $o_i \neq y_i$ then

　　　　if $o_i = 0$ then for $j = 1$ to n

　　　　　$w_{ij} = w_{ij} + x_i$

　　　　else for $j = 1$ to n

　　　　　$w_{ij} = w_{ij} - x_i$

在算法中,依次对输出层的每一个神经元的理想输出和实际输出进行比较。如果它们不相

同，则对相应的联接权进行修改。相当于将对离散单输出感知器的神经元的处理逐个地用于离散多输出感知器输出层的每一个神经元。

在算法3－1和3－2中，第1步，要求对神经元的联接权进行初始化。在程序的实现中，就是给 W 一个初值。实验表明，对大部分网络模型来说（也有例外情况），W 的各个元素不能用相同的数据进行初始化，因为这样会使网络失去学习能力。一般地，使用一系列小伪随机数对 W 进行初始化。

第2步的控制是说"重复下列过程，直到训练完成"。但是，什么情况下为"训练完成"呢？一般地，很难对每个样本重复一次就可以达到精度要求，算法必须经过多次的迭代，才有可能使网络的精度达到要求。问题是如何来控制这个迭代次数。一种方法是对样本集执行规定次数的迭代，另一种方法是给定一个精度控制参数，第三种方法是将这两种方法结合起来使用。这里所说的精度是指网络的实际输出与理想输出之间的差别。

对第一种方法，可以采用如下方式加以实现：设置一个参数，用来纪录算法的迭代次数。同时，在程序中设定一个最大循环次数的值。当迭代次数未达到该值时，迭代继续进行；当迭代次数超过此值时，迭代停止。该方法存在的问题是，对一个给定的样本集，事先并不知道究竟需要迭代多少次，网络的精度才可以达到用户的要求。迭代的次数太多，会损失训练算法的效率；迭代的次数太少，网络的精度就难以达到用户的要求。因此，仅仅用迭代的次数实施对网络训练的控制是难以取得令人满意的结果的。一种改进的方法是采用分阶段进行迭代的控制方法：设定一个基本的迭代次数 N，每当训练完成 N 次迭代后，就给出一个中间结果。如果此中间结果满足要求，则停止训练；否则，将进行下一个 N 次迭代训练。如此下去，直到训练完成。当然，这样做需要程序能够实现训练的暂停、继续、停止等控制。

第二种方法的实现与第一种方法的实现类似，只是比较的对象不同罢了。这种方法要解决的问题有两个：首先，要解决精度的度量问题，最简单的方法是用所有样本的实际输出向量与理想输出向量的对应分量的差的绝对值之和作为误差的度量。另一种简单的方法是用所有样本的实际输出向量与理想输出向量的欧氏距离的和作为误差的度量。一般地，用户可以根据实际问题，选择一个适当的度量。其次，存在这样的可能，网络无法表示样本所代表的问题。在这种情况下，网络在训练中可能总也达不到用户的精度要求。这时，训练可能成为"死循环"。

为了用这两种方法各自的优点去弥补对方的缺点，可以将这两种方法结合起来综合使用，构成第三种方法。也就是同时使用迭代次数和精度来实现训练控制。

在精度控制中，用户需要首先根据实际问题，给定一个训练精度控制参数。建议在系统初始测试阶段，这个精度要求可以低一些，测试完成后，再给出实际的精度要求。这样做的目的是，避免在测试阶段花费太多时间，因为有时训练时间是很长的。

还需要指出的是，在算法的实现中，读者还可以采用一些方法，从不同的角度去提高算法的效率。

3.2.3 连续多输出感知器训练算法

在掌握了感知器的基本训练算法后，现在将感知器中各神经元的输出函数改成非阶跃函数，

使它们的输出值变成是连续性的，从而使得网络的输入、输出向量更具一般性，更容易适应应用的要求，达到较好地扩充网络的功能和应用范围的目的。由于只是网络的神经元的激活函数发生了变化，其拓扑结构仍然不变，所以在下面讨论连续多输出感知器训练算法时，仍然参考图 3－2，$\boldsymbol{W}$、$\boldsymbol{O}$、$\boldsymbol{X}$、$\boldsymbol{Y}$、n、m 等参数的意义如上所述。在下面给出的算法 3－3 中，我们使用上面提到的第二种方法来实现对迭代次数的控制。ε 被用来表示训练的精度要求。不同的是，在这里，$\boldsymbol{X}$、$\boldsymbol{Y}$ 的分量的值可以是一般的实数。

算法 3－3　连续多输出感知器训练算法

1　用适当的小伪随机数初始化权矩阵 $\boldsymbol{W}$；

2　初置精度控制参数 ε、学习率 α、精度控制变量 $d=\varepsilon+1$；

3　While $d \geqslant \varepsilon$ do

　3.1　$d=0$；

　3.2　for 每个样本$(\boldsymbol{X},\boldsymbol{Y})$do

　　3.2.1　输入 $\boldsymbol{X}(=(x_1,x_2,\cdots,x_n))$；

　　3.2.2　求 $\boldsymbol{O}=\boldsymbol{F}(\boldsymbol{XW})$；

　　3.2.3　修改权矩阵 $\boldsymbol{W}$：

　　　for $i=1$ to n，$j=1$ to m do

　　　　$w_{ij}=w_{ij}+\alpha(y_j-o_j)x_i$；

　　3.2.4　累积误差

　　　for $j=1$ to m do

　　　　$d=d+(y_j-o_j)^2$

在上述算法中，用公式 $w_{ij}=w_{ij}+\alpha(y_j-o_j)x_i$ 取代了算法 3－2 第 2.1.3 步中的多个判断。y_j 与 o_j 之间的差别对 w_{ij} 的影响由 $\alpha(y_j-o_j)x_i$ 表现出来。这样处理之后，不仅使得算法的控制在结构上更容易理解，而且还使得它的适应面更宽。

当用计算机程序实现该算法时，ε、α、d、i、j、n、m 均可以用简单变量来表示，$\boldsymbol{W}$ 可以用一个 n 行 m 列的二维数组存放。建议将样本集用两个二维数组存放，一个 p 行 n 列的二维数组用来存放输入向量集，它的每一行表示一个输入向量；另一个 p 行 m 列的二维数组用来存放相应的理想输出向量集，它的每一行表示一个对应的理想输出向量。读者也可以根据自己的习惯确定存放这些数据的方式。

另外，在系统的调试过程中，可以在适当的位置加入一些语句，用来显示网络目前的状态。如按一定的间隔显示实际输出向量与理想输出向量的比较、联接矩阵、误差测度等，使得系统的调试过程可以在设计者/调试者的良好控制下进行。当然，根据需要，也可以将这些数据以文件的形式存放起来，以便于过后进行更深入的分析。

实际上，上面给出的算法只是一些基本算法，在实现过程中，读者还可以对它们进行适当的修改，以使它们有更高的运行效率，并能获得更好的效果。

上述感知器的训练算法，还有一些值得注意的问题。

一方面，Minsky 曾经在 1969 年证明，有许多基本问题是感知器无法解决的，这类问题被称为线性不可分问题。所以，算法遇到的第一个问题是，样本集所代表的问题是否是线性可分的？由于抽样的随机性，有时甚至会出现这样的现象：问题本身是线性可分的，但样本集反映出来的却是线性不可分的，或者相反。这虽然是抽样的技术问题，但在实际上是存在的。因此，也应该引起足够的重视。将这一问题作为困扰感知器的第一个问题。

第二，由于世界是在不断变化的，所以，一个问题可能在某一时刻是线性可分的，而在另一时刻又变得线性不可分。这类问题的处理就更为困难了。

第三，由于问题是通过抽样得来的实际数据表示的，它很可能不是我们习惯的数据模型的表现形式。所以，很难直接从样本数据集看出该问题是否是线性可分的。

此外，未能证明，一个感知器究竟需要经过多少步才能完成训练。而且，给出的算法是否优于穷举法，也是未能说明的。在简单情况下，穷举法可能会更好。

显然，上述问题都是与样本集相关的。这就相当于说，问题（指被模拟的系统）本身对感知器的影响是非常大的。下面，来考虑线性不可分问题。

3.3 线性不可分问题

由 Rossenblatt 所给出的感知器的学习定理表明，感知器可以学会它所能表达的任何东西。与人类的大脑相同，表达能力和学习能力是不同的。“表达”是指感知器模拟特殊功能的能力，而学习要求由一个用于调整联接权以产生具体表示的一个过程的存在。显然，如果感知器不能够表达相应的问题，就无从考虑它是否能够学会该问题了。所以，这里的“它能表示”成为问题的关键。也就是说，是否存在一些问题，它们不能被感知器表示呢？

前面已经提到，Minsky 在 1969 年就指出，感知器甚至无法解决像“异或”这样简单的问题。那么，这类问题是什么样的问题？它有什么特点？除了“异或”，还有多少这样的问题？这样的问题是有限的么？下面从“异或”问题入手进行相应的分析，希望找出这一类问题的特性来，以寻找相应的解决方法。

3.3.1 异或(Exclusive - OR)问题

Minsky 得出的最令世人失望的结果是：感知器无法实现最基本的“异或”运算。而“异或”运算是电子计算机最基本的运算之一。这就预示着人工神经网络将无法解决电子计算机可以解决的大量的问题。因此，它的功能是极为有限的，是没有前途的。那么，感知器为什么无法解决“异或”问题呢？先看“异或”运算的定义：

$$g(x,y)=\begin{cases}0 & \text{if } x=y\\ 1 & \text{其他}\end{cases}$$

相应的真值表如表 3-1 所示。

表 3-1 异或运算的真值表

$g(x,y)$		运算对象 y	
		0	1
运算对象 x	0	0	1
	1	1	0

由定义可知，这是一个双输入、单输出的问题。也就是说，如果感知器能够表达它，则此感知器输入应该是一个二维向量，输出则为标量。因此，该感知器可以只含有一个神经元。为方便起见，设输入向量为$(\boldsymbol{x},\boldsymbol{y})$，输出为 $\boldsymbol{o}$，神经元的阈值为 θ。感知器如图 3-3 所示，图 3-4 所示是网络函数的图像。显然，无论如何选择 a、b、θ 的值，都无法使得直线将点(0,0)和点(1,1)(它们对应的函数值为 0)与点(0,1)和点(1,0)(它们对应的函数值为 1)划分开来。即使使用 s 形函数，也难以做到这一点。这种由单级感知器不能表达的问题被称为线性不可分问题。



图 3-3 单神经元感知器　　　　图 3-4 平面划分

有了上述思路，来考察只有两个自变量且自变量只取 0 或 1 的函数的基本情况。表 3-2 给出了所有这种函数的定义。其中，f_7、f_{10}为线性不可分的，其他均为线性可分的。不过，当变量的个数较多时，难以找到一个较简单的方法去确定一个函数是否为线性可分的。事实上，这种线性不可分的函数随着变量个数的增加而快速增加，甚至远远超过了线性可分函数的个数。现在，我们仍然只考虑二值函数的情况。设函数有 n 个自变量，因为每个自变量的值只可以取 0 或 1，从而函数共有 2^n 个输入模式。在不同的函数中，每个模式的值可以为 0 或者 1。这样，我们总共可以得到 2^{2^n}种不同的函数。表 3-3 是 R. O. Windner 1960 年给出的 n 为 1～6 时二值函数的个数以及其中的线性可分函数的个数的研究结果。从中我们看出，当 n 大于等于 4 时，线性不可分函数的个数远远大于线性可分函数的个数。而且随着 n 的增大，这种差距会在数量级上越来越大。这表明，感知器不能表达的问题的数量远远超过了它所能表达的问题的数量。这也难怪当 Minsky 给出感知器的这一致命缺陷时，会使人工神经网络的研究跌入漫长的黑暗期。

表 3-2 含两个自变量的所有二值函数

自变量		函数及其值															
x	y	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

表 3-3　二值函数与线性可分函数的个数

自变量个数	函数的个数	线性可分函数的个数
1	4	4
2	16	14
3	256	104
4	65 536	1 882
5	4.3×10^{9}	94 572
6	1.8×10^{19}	5 028 134

3.3.2　线性不可分问题的克服

到了20世纪60年代后期，人们就弄清楚了线性不可分问题，并且知道，单级网的这种限制可以通过增加网络的层数来解决。

事实上，一个单级网络可以将平面划分成两部分，用多个单级网组合在一起，并用其中的一个去综合其他单级网的结果，就可以构成一个两级网络，该网络可以被用来在平面上划分出一个封闭或者开放的凸域来。如图3-5所示，如果第1层含有 n 个神经元，则每个神经元可以确定一条 n 维空间中的直线，其中，AN_i 用来确定第 i 条边。输出层的 AN_o 用来实现对它们的综合。这样，就可以用一个两级单输出网在 n 维空间中划分出一个 m 边凸域来。在这里，图中第2层的神经元相当于一个与门。当然，根据实际需要，输出层的神经元可以有多个。这可以根据网络要模拟的实际问题来决定。

按照这些分析，很容易构造出第一层含两个神经元，第二层含一个神经元的两级网络来实现"异或"运算。

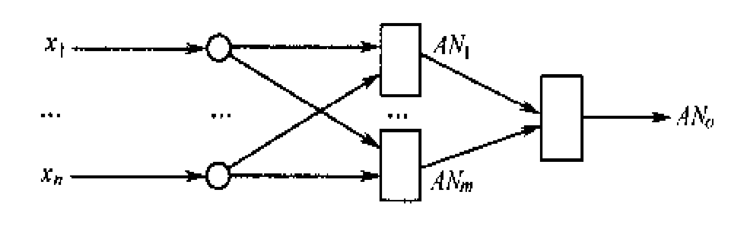


图3-5　两级单输出网

当然，输出层的神经元可以不仅仅实现"与"运算，它也可以实现其他类型的函数。

此外，网络的输入输出也可以是非二值的，这样，网络识别出来的就是一个连续的域，而不仅仅是域中的有限个离散的点。

一个非凸域可以拆分成多个凸域。因此，三级网将会更一般一些，可以用它去识别出一些非凸域来。而且在一定的范围内，网络所表现出来的分类功能主要受到神经元的个数和各个联接

权的限制。这些问题显然又是与问题紧密相关的。

多级网络虽然很好地解决了线性不可分问题，但是，由于无法知道网络隐藏层的神经元的理想输出，所以，感知器的训练算法是难以直接用于多层网的训练。因此，在多级网训练算法的设计中，解决好隐藏层的联接权的调整问题是非常关键的。

练 习 题

1. 叙述感知器的体系结构，并指出输入/输出向量的实际意义。

2. 请用你熟悉的一种计算机语言实现算法 3-1，并用你的实现完成对 0,1,2,3,…,9 的奇偶性的识别(提示，请首先给出这 10 个数字的适当表示)。

3. 请用你熟悉的一种计算机语言实现算法 3-2，在实现中，注意对本章所给的两种精度控制方法的实验。

4. 请用你熟悉的一种计算机语言实现算法 3-3。

5. 对你实现的算法 3-3，寻找两个不同的样本集，它们一个使训练收敛，另一个使训练发散。

6. 什么叫线性不可分问题？早期的感知器为什么不能表示线性不可分问题？

7. 用多级网络可以克服线性不可分问题，而多级网络似乎是"简单地"由多个单级网络"串接"而成的，但人工神经网络的研究在这段路上却走了许多年。你认为，困难在何处？你有克服这个困难的思路么？

第四章 BP 网 络

在上一章介绍的感知器的算法中,理想输出与实际输出之差被用来估计直接到达该神经元的联接的权重的误差。当为解决线性不可分问题而引入多级网络后,如何估计网络隐藏层的神经元的误差就成了难题。因为在实际中,无法知道隐藏层的任何神经元的理想输出值。BP(Back propagation)算法在于利用输出层的误差来估计输出层的直接前导层的误差,再用这个误差估计更前一层的误差。如此下去,就获得了所有其他各层的误差估计。这样就形成了将输出端表现出的误差沿着与输入信号传送相反的方向逐级向网络的输入端传递的过程。因此,人们将此算法称为向后传播算法,简称 BP 算法。使用 BP 算法进行学习的多级非循环网络称为 BP 网络。虽然这种误差估计本身的精度会随着误差本身的"向后传播"而不断降低,但它还是给多层网络的训练提供了较有效的办法。所以,多年来该算法受到了广泛的关注。

本章将介绍 BP 网络的构成及其训练过程;隐藏层权调整方法的直观分析,BP 训练算法中使用的 Delta 规则(最速下降法)的理论推导;算法的收敛速度及其改进讨论;BP 网络中的几个重要问题。

4.1 概 述

BP 算法是非循环多级网络的训练算法。虽然该算法的收敛速度非常慢,但由于它具有广泛的适用性,使得它在 1986 年被提出后,很快就成为应用最为广泛的多级网络训练算法,并对人工神经网络的推广应用发挥了重要作用。

人们公认,BP 算法对人工神经网络的第二次研究高潮的到来起到了很大的作用。从某种意义上讲,BP 算法的出现,结束了多层网络没有训练算法的历史,并被认为是多级网络系统的训练方法。此外,它还有很强的数学基础,所以,其联接权的修改是令人信服的。

但是,BP 算法也有它的弱点。非常慢的训练速度、高维曲面上局部极小点的逃离问题、算法的收敛问题等都是困扰 BP 网络的严重问题,尤其是后面的两个问题,甚至会导致网络的失败。虽然它有这样一些限制,并且有许多难以令人满意的地方,但是,其广泛的适应性和有效性使得人工神经网络的应用范围得到了较大的扩展。

从 BP 算法被重新发现到引起人们的广泛关注并发挥巨大的作用,应该归功于 UCSD 的 PDP(Parallel Distributed Processing)研究小组的 Rumelhart、Hinton 和 Williams。他们在 1986 年独立地给出了 BP 算法清楚而简单的描述,使得该算法非常容易让人掌握并加以实现。另外,由于此时人们对人工神经网络的研究正处于第二高潮期,而且 PDP 小组在人工神经网络上的丰富的研究成果也给其发表能受到广泛的关注提供了便利条件。在该成果发表后不久人们就发现,

早在 1982 年 Paker 就完成了相似的工作。后来人们进一步地发现，甚至在更早的 1974 年，Werbos 就已提出了该方法的描述。遗憾的是，无论是 Paker，还是 Werbos，他们的工作在完成并发表十余年后，都没能引起人们的关注，这无形中导致了多级网络的训练算法及其推广应用向后推迟了十余年。通过这件事情，也应该看到，要想使重要的研究成果能引起广泛的重视而尽快发挥作用，论文的发表也是非常重要的。

4.2 基本 BP 算法

4.2.1 网络的构成

1. 神经元

与一般的人工神经网络一样，构成 BP 网的神经元仍然是 2.2 节中定义的神经元。按照 BP 算法的要求，这些神经元所用的激活函数必须是处处可导的。一般地，多数设计者都使用 s 形函数。对一个神经元来说，取它的网络输入

$$net = x_1w_1 + x_2w_2 + \cdots + x_nw_n$$

其中，$x_1, x_2, \cdots, x_n$ 为该神经元所接受的输入，$w_1, w_2, \cdots, w_n$ 分别是它们对应的联接权。该神经元的输出为

$$o = f(net) = \frac{1}{1 + e^{-net}} \tag{4-1}$$

其相应的图像如图 4-1 所示。当 $net = 0$ 时，o 取值为 0.5，并且 net 落在区间 $(-0.6, 0.6)$ 中时，o 的变化率比较大，而在 $(-1,1)$ 之外，o 的变化率就非常小。

现求 o 关于 net 的导数：

$$\begin{aligned} f'(net) &= \frac{e^{-net}}{(1 + e^{-net})^2} = \frac{1 + e^{-net} - 1}{(1 + e^{-net})^2} \\ &= \frac{1}{1 + e^{-net}} - \frac{1}{(1 + e^{-net})^2} \\ &= o - o^2 = o(1 - o) \end{aligned}$$

注意到：

$$\lim_{net \to +\infty} \frac{1}{1 + e^{-net}} = 1, \quad \lim_{net \to -\infty} \frac{1}{1 + e^{-net}} = 0$$

根据式 4-1 可知，o 的值域为 $(0,1)$，从而，$f'(net)$ 的值域为 $(0, 0.25)$，而且是在 o 为 0.5 时，$f'(net)$ 达到极大值。其图像如图 4-2 所示。

请读者注意，图 4-1、4-2 告诉我们，在后续对训练的讨论中，应该将 net 的值尽量控制在收敛比较快的范围内。

实际上，也可以用其他函数作为 BP 网络神经元的激活函数，只要该函数是处处可导的。

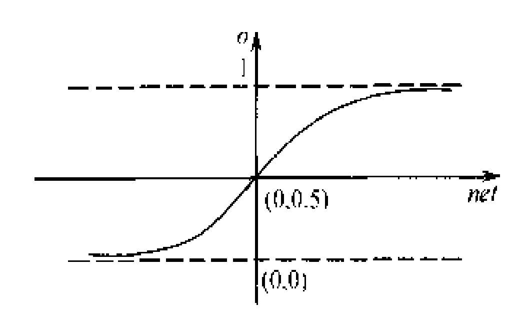



图 4-1　BP 网的神经元的激活函数的图像

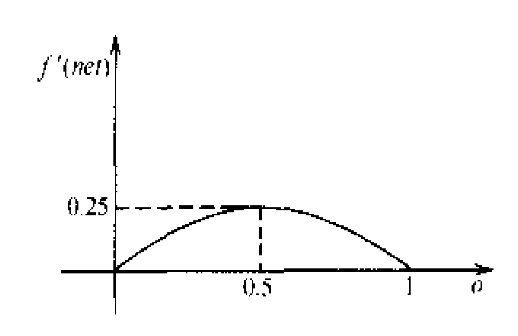


图 4-2　$f'(net)$的图像

2. 网络的拓扑结构

BP 网的结构如图 2-7 所示。实际上，只需用一个二级网络，就可以说明 BP 算法。一般地，设 BP 网络的输入样本集为

$$\{(\boldsymbol{X},\boldsymbol{Y}) \quad \boldsymbol{X} \text{ 为输入向量}, \boldsymbol{Y} \text{ 为 } \boldsymbol{X} \text{ 对应的理想输出向量}\}$$

网络有 n 层，第 $h(1\leqslant h\leqslant n)$层神经元的个数用 L_h 表示，该层神经元的激活函数用 F_h 表示，该层的联接矩阵用 $\boldsymbol{W}^{(h)}$表示。

显然，输入向量、输出向量的维数是由问题所直接决定的，然而，网络隐藏层的层数和各个隐藏层神经元的个数则是与问题相关的。目前的研究结果还难以给出它们与问题的类型及其规模之间的函数关系。实验表明，增加隐藏层的层数和隐藏层神经元的个数不一定能够提高网络的精度和表达能力，在多数情况下，BP 网一般都选用二级网络。

4.2.2　训练过程概述

首先，前面我们已经提到过，人工神经网络的训练过程是根据样本集对神经元之间的联接权进行调整的过程，BP 网络也不例外。其次，BP 网络执行的是有导师训练。所以，其样本集是由形如：

（输入向量，理想输出向量）

的向量对构成的。所有这些向量对，都应该是来源于网络即将模拟的系统的实际“运行”结果。它们可以是从实际运行系统中采集来的。

在开始训练前，所有的权都应该用一些不同的小随机数进行初始化。“小随机数”用来保证网络不会因为权过大而进入饱和状态，从而导致训练失败；“不同”用来保证网络可以正常地学习。实际上，如果用相同的数去初始化权矩阵，则网络将无能力学习。

BP 算法主要包含 4 步，这 4 步被分为两个阶段：

1. 向前传播阶段

(1) 从样本集中取一个样本$(\boldsymbol{X}_p,\boldsymbol{Y}_p)$，将 $\boldsymbol{X}_p$ 输入网络；

(2) 计算相应的实际输出 $\boldsymbol{O}_p$。

在此阶段，信息从输入层经过逐级的变换，传送到输出层。这个过程也是网络在完成训练后正常运行时执行的过程。在此过程中，网络执行的是下列运算：

$$O_p = F_n(\ldots(F_2(F_1(X_p W^{(1)})W^{(2)})\cdots)W^{(n)})$$

2. 向后传播阶段

(1) 计算实际输出 O_p 与相应的理想输出 Y_p 的差；

(2) 按极小化误差的方式调整权矩阵。

这两个阶段的工作一般应受到精度要求的控制，在这里，取

$$E_p = \frac{1}{2}\sum_{j=1}^{m}(y_{pj} - o_{pj})^2 \qquad 4-2$$

作为网络关于第 p 个样本的误差测度。而将网络关于整个样本集的误差测度定义为

$$E = \sum E_p \qquad 4-3$$

如前所述，之所以将此阶段称为向后传播阶段，是对应于输入信号的正常传播而言的。因为在开始调整神经元的联接权时，只能求出输出层的误差，而其他层的误差要通过此误差反向逐层后推才能得到。有时候也称之为误差传播阶段。

4.2.3 误差传播分析

1. 输出层权的调整

为了说明清晰方便，我们使用图 4-3 中的相应符号来讨论输出层联接权的调整。

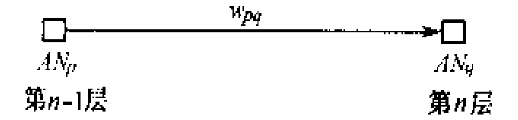


图 4-3 AN_p 到 AN_q 的联接

图中，AN_q 是输出层的第 q 个神经元，w_{pq}是从其前导层的第 p 个神经元到AN_q 的联接权。取

$$w_{pq} = w_{pq} + \Delta w_{pq} \qquad 4-4$$

根据第 2 章给出的 Delta 规则(式 2-20)，有

$$\Delta w_{pq} = \alpha\delta_q o_p \qquad 4-5$$

由于在本书中不再区分神经元的激活状态和输出值，所以上式中的 δ_q 的计算按下式进行：

$$\delta_q = f'_n(net_q)(y_q - o_q) \qquad 4-6$$

而

$$f'_n(net_q) = o_q(1 - o_q)$$

所以

$$\begin{aligned}\Delta w_{pq} &= \alpha\delta_q o_p \\ &= \alpha f'_n(net_q)(y_q - o_q)o_p \\ &= \alpha o_q(1 - o_q)(y_q - o_q)o_p\end{aligned}$$

即

$$\Delta w_{pq} = \alpha o_q(1 - o_q)(y_q - o_q)o_p \qquad 4-7$$

δ_q 可以看成是 AN_q 所表现出来的误差。它由 AN_q 的输出值和 AN_q 的理想输出值以及与 w_{pq} 直接相关联的 AN_p 的输出值确定。

2. 隐藏层权的调整

对隐藏层权的调整,仍然可以采用式 4－4、4－5,只不过在这里不再可以用式 4－6 去计算相应的神经元所表现出来的误差,因为此时无法知道该神经元的理想输出。为了解决这个问题,在这里先从直观上来研究如何计算相应的神经元所表现出来的误差,相应的理论推导将留在 4.6 节给出。

为使讨论更清晰,对隐藏层联接权调整的讨论将参考图 4－4 进行。按照该图的表示,省去了其中有些符号上表示网络层号的上标。一方面,将相应的层号标注在图的下方。另一方面,仅在需要的地方让层号以下标的形式出现。

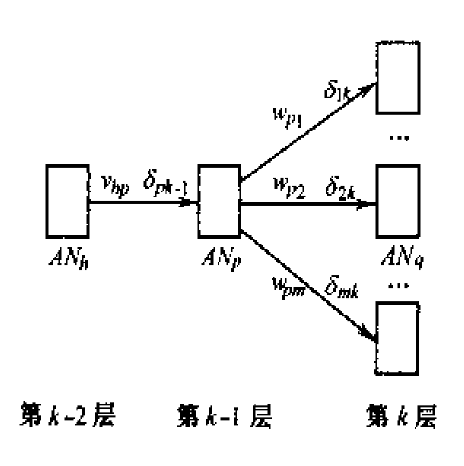


图 4－4　误差反向传播示意图

假定图 4－4 中的 $w_{p1},w_{p2},\cdots,w_{pm}$ 的调整已经完成。所以,此时 $\delta_{1k},\delta_{2k},\cdots,\delta_{mk}$ 的值是已知的。要想调整 v_{hp},就必须知道 δ_{pk-1},由于 AN_p 的理想输出是未知的,所以,必须按照一定的方法来给 δ_{pk-1} 一个合适的估计。

从图中可以看出,δ_{pk-1} 的值应该是和 $\delta_{1k},\delta_{2k},\cdots,\delta_{mk}$ 有关的,在 $\delta_{1k},\delta_{2k},\cdots,\delta_{mk}$ 等每个值中,都含有 δ_{pk-1} 的"成分"。因此,自然地想到用 $\delta_{1k},\delta_{2k},\cdots,\delta_{mk}$ 来估计 δ_{pk-1}。同时,δ_{pk-1} 又是通过 $w_{p1},w_{p2},\cdots,w_{pm}$ 与 $\delta_{1k},\delta_{2k},\cdots,\delta_{mk}$ 关联的。具体地,不妨认为 δ_{pk-1}:

通过权 w_{p1} 对 δ_{1k} 做出贡献,

通过权 w_{p2} 对 δ_{2k} 做出贡献,

……

通过权 w_{pm} 对 δ_{mk} 做出贡献。

从而,AN_p 的输出误差是与

$$w_{p1}\delta_{1k}+w_{p2}\delta_{2k}+\cdots+w_{pm}\delta_{mk}$$

相关的。这样,可以用它近似地表示 AN_p 的理想输出与实际输出的差。根据式 4－6 得到:

$$\delta_{pk-1}=f'_{k-1}(net_p)(w_{p1}\delta_{1k}+w_{p2}\delta_{2k}+\cdots+w_{pm}\delta_{mk}) \qquad 4-8$$

从而有:

$$
\begin{aligned}
\Delta v_{hp} &= \alpha\delta_{pk-1}o_{hk-2} \\
&= \alpha f'_{k-1}(net_p)(w_{p1}\delta_{1k}+w_{p2}\delta_{2k}+\cdots+w_{pm}\delta_{mk})\,o_{hk-2} \\
&= \alpha o_{pk-1}(1-o_{pk-1})\,(w_{p1}\delta_{1k}+w_{p2}\delta_{2k}+\cdots+w_{pm}\delta_{mk})o_{hk-2}
\end{aligned}
$$

即：

$$
\Delta v_{hp} = \alpha o_{pk-1}(1-o_{pk-1})\,(w_{p1}\delta_{1k}+w_{p2}\delta_{2k}+\cdots+w_{pm}\delta_{mk})o_{hk-2} \qquad 4-9
$$

$$
v_{hp} = v_{hp} + \Delta v_{kp} \qquad 4-10
$$

式中，o_{pk-1}、o_{hk-2}分别表示第 $k-1$ 层的第 p 个神经元、第 $k-2$ 层的第 h 个神经元的输出。

4.2.4 基本的BP算法

知识的分布表示原理指出，由于知识是分布表示的，所以人工神经网络可以在实际应用中根据不断获取的经验来增加自己的处理能力。因此，它的学习可以不是一次完成的。也就是说，人工神经网络应该可以在工作过程中通过对新样本的学习而获得新的知识，以不断丰富自己的知识。这就要求在一定的范围内，网络在学会新知识的同时，保持原来学会的东西不被忘记。这个特性被称为可塑性。

然而，BP 网络并不具有这种可塑性。它要求用户一开始就要将所有要学的样本一次性地交给它，而不是"学会"一个以后，再学其他的。这就要求我们不能在完成一个样本的训练后才进行下一个样本的训练。所以，训练算法的最外层循环应该是"精度要求"，其次才是对样本集中的样本进行循环。也就是，在 BP 网络针对一个样本对各个联接权作一次调整后，虽然此样本还不能满足精度要求，此时也不能继续按此样本进行训练，而应考虑其他的样本，待样本集中的所有的样本都被考虑过一遍后，再重复这个过程，直到网络能同时满足各个样本的要求。

具体做法是，对样本集

$$
\boldsymbol{S} = \{(\boldsymbol{X}_1,\ \boldsymbol{Y}_1),(\boldsymbol{X}_2,\ \boldsymbol{Y}_2),\cdots,(\boldsymbol{X}_s,\ \boldsymbol{Y}_s)\}
$$

网络根据$(\boldsymbol{X}_1,\ \boldsymbol{Y}_1)$计算出实际输出 $\boldsymbol{O}_1$ 和误差测度 E_1，对 $\boldsymbol{W}^{(1)},\boldsymbol{W}^{(2)},\cdots,\boldsymbol{W}^{(M)}$各做一次调整；在此基础上，再根据$(\boldsymbol{X}_2,\boldsymbol{Y}_2)$计算出实际输出 $\boldsymbol{O}_2$ 和误差测度 E_2，对 $\boldsymbol{W}^{(1)},\boldsymbol{W}^{(2)},\cdots,\boldsymbol{W}^{(M)}$分别做第二次调整……如此下去。本次循环最后再根据$(\boldsymbol{X}_s,\boldsymbol{Y}_s)$计算出实际输出 $\boldsymbol{O}_s$ 和误差测度 E_s，对 $\boldsymbol{W}^{(1)},\boldsymbol{W}^{(2)},\cdots,\boldsymbol{W}^{(M)}$分别做第 s 次调整。这个过程，相当于是对样本集中各个样本的一次循环处理。这个循环需要重复下去，直到对整个样本集来说，误差测度的总和满足系统的要求为止，即：

$$
\sum E_p < \varepsilon
$$

这里，ε 为精度控制参数。按照这一处理思想，可以得出下列基本的 BP 算法：

算法 4-1　基本 BP 算法

1　for $h=1$ to M do

　1.1　初始化 $\boldsymbol{W}^{(h)}$；

2　初始化精度控制参数 ε；

3 $E = \varepsilon + 1$;

4 while $E > \varepsilon$ do

  4.1 $E = 0$;

  4.2 对 S 中的每一个样本$(\boldsymbol{X}_p, \boldsymbol{Y}_p)$:

    4.2.1 计算出 $\boldsymbol{X}_p$ 对应的实际输出 $\boldsymbol{O}_p$;

    4.2.2 计算出 E_p;

    4.2.3 $E = E + E_p$;

    4.2.4 根据式 4-4、4-7 调整 $\boldsymbol{W}^{(M)}$;

    4.2.5 $h = M - 1$;

    4.2.6 while $h \neq 0$ do

      4.2.6.1 根据式 4-9、4-10 调整 $\boldsymbol{W}^{(h)}$;

      4.2.6.2 $h = h - 1$

  4.3 $E = E/2.0$

4.3 算法的改进

实验表明,算法 4-1 较好地抽取了样本集中所含的输入向量和输出向量之间的关系。通过对实验结果的仔细分析会发现,BP 网络接受样本的顺序仍然对训练的结果有较大的影响。比较而言,它更"偏爱"较后出现的样本:如果每次循环都按照$(\boldsymbol{X}_1, \boldsymbol{Y}_1)$,$(\boldsymbol{X}_2, \boldsymbol{Y}_2)$,…,$(\boldsymbol{X}_s, \boldsymbol{Y}_s)$所给定的顺序进行训练,在网络"学成"投入运行后,对于与该样本序列较后的样本较接近的输入,网络所给出的输出的精度将明显高于与样本序列较前的样本较接近的输入对应的输出的精度。那么,是否可以根据样本集的具体情况,给样本集中的样本安排一个适当的顺序,以求达到基本消除样本顺序的影响,获得更好的学习效果呢?这是非常困难的。因为无论我们如何排列这些样本,它终归要有一个顺序,序列排得好,顺序的影响只会稍微小一些。另外,要想给样本数据排定一个顺序,本来就不是一件容易的事情,再加上要考虑网络本身的因素,就更困难了。

样本顺序对结果的影响的原因是什么呢?深入分析算法 4-1 可以发现,造成样本顺序对结果产生严重影响的原因是:算法对 $\boldsymbol{W}^{(1)}$,$\boldsymbol{W}^{(2)}$,…,$\boldsymbol{W}^{(M)}$的调整是分别依次根据$(\boldsymbol{X}_1, \boldsymbol{Y}_1)$,$(\boldsymbol{X}_2, \boldsymbol{Y}_2)$,…,$(\boldsymbol{X}_s, \boldsymbol{Y}_s)$完成的。"分别"、"依次"决定了网络对"后来者"的"偏爱"。实际上,按照这种方法进行训练,有时甚至会引起训练过程的严重抖动,更严重的,它可能使网络难以达到用户要求的训练精度。这是因为排在较前的样本对网络的部分影响被排在较后的样本的影响掩盖掉了,从而使排在较后的样本对最终结果的影响就要比排在较前的样本的影响大。这又一次表明,虽然知识的分布表示原理告诉我们,信息的局部破坏不会对原信息产生致命的影响,但是这个被允许的破坏是非常有限的。此外,算法在根据后来的样本修改网络的联接矩阵时,进行的是全面的修改,这使得"信息的破坏"也变得不再是局部的。这正是 BP 网络在遇到新内容时,必须重新对整个样本集进行学习的主要原因。

虽然在精度要求不高的情况下，顺序的影响有时是可以忽略的，但是我们还是应该尽量地消除它。那么，如何消除样本顺序对结果的影响呢？根据上述分析，算法应该避免“分别”、“依次”的出现。因此，我们不再“分别”、“依次”根据$(\boldsymbol{X}_1, \boldsymbol{Y}_1), (\boldsymbol{X}_2, \boldsymbol{Y}_2), \cdots, (\boldsymbol{X}_s, \boldsymbol{Y}_s)$对 $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \cdots, \boldsymbol{W}^{(M)}$进行调整，而是用$(\boldsymbol{X}_1, \boldsymbol{Y}_1), (\boldsymbol{X}_2, \boldsymbol{Y}_2), \cdots, (\boldsymbol{X}_s, \boldsymbol{Y}_s)$的“总效果”去实施对 $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \cdots, \boldsymbol{W}^{(M)}$的修改。这就可以较好地将对样本集的样本的一系列学习变成对整个样本集的学习。获取样本集“总效果”的最简单的办法是取

$$\Delta w_{ij}^{(h)} = \sum \Delta_p w_{ij}^{(h)} \qquad 4-11$$

其中的$\sum$表示对整个样本集的求和，$\Delta_p w_{ij}^{(h)}$代表联接权$w_{ij}^{(h)}$关于样本$(\boldsymbol{X}_p, \boldsymbol{Y}_p)$的调整量。从而得到算法 4－2。

算法 4－2　消除样本顺序影响的 BP 算法

1　for $h = 1$ to M do
　1.1　初始化 $\boldsymbol{W}^{(h)}$；
2　初始化精度控制参数 ε；
3　$E = \varepsilon + 1$；
4　while $E > \varepsilon$ do
　4.1　$E = 0$；
　4.2　对所有的 i、j、h：$\Delta w_{ij}^{(h)} = 0$；
　4.3　对 $\boldsymbol{S}$ 中的每一个样本$(\boldsymbol{X}_p, \boldsymbol{Y}_p)$：
　　4.3.1　计算出 $\boldsymbol{X}_p$ 对应的实际输出 $\boldsymbol{O}_p$；
　　4.3.2　计算出 E_p；
　　4.3.3　$E = E + E_p$；
　　4.3.4　对所有的 i、j 根据式 4－7 计算 $\Delta_p w_{ij}^{(M)}$；
　　4.3.5　对所有的 i, j：$\Delta w_{ij}^{(M)} = \Delta w_{ij}^{(M)} + \Delta_p w_{ij}^{(M)}$；
　　4.3.6　$h = M - 1$；
　　4.3.7　while $h \neq 0$ do
　　　4.3.7.1　对所有的 i、j 根据式 4－9 计算 $\Delta_p w_{ij}^{(h)}$；
　　　4.3.7.2　对所有的 i、j：$\Delta w_{ij}^{(h)} = \Delta w_{ij}^{(h)} + \Delta_p w_{ij}^{(h)}$；
　　　4.3.7.3　$h = h - 1$
　4.4　对所有的 i、j、h：$w_{ij}^{(h)} = w_{ij}^{(h)} + \Delta w_{ij}^{(h)}$；
　4.5　$E = E/2.0$

上述算法较好地解决了因样本的顺序引起的精度问题和训练的抖动问题。但是，该算法的收敛速度还是比较慢的。为了解决收敛速度问题，人们也对算法进行了适当的改造。例如：给每一个神经元增加一个偏移量来加快收敛速度；直接在激活函数上加一个位移使其避免因获得 0

输出而使相应的联接权失去获得训练的机会;联接权的本次修改要考虑上次修改的影响,以减少抖动问题。Rumelhart 等人 1986 年提出的考虑上次修改的影响的公式为:

$$\Delta w_{ij} = \alpha\delta_j o_i + \beta\Delta w'_{ij}$$

其中,$\Delta w'_{ij}$为上一次的修改量,β 为冲量系数,一般可取到 0.9。1987 年,Sejnowski 与 Rosenberg 给出了基于指数平滑的方法,它对某些问题是非常有效的:

$$\Delta w_{ij} = \alpha((1-\beta)\delta_j o_i + \beta\Delta w'_{ij}) \qquad 4-13$$

其中,$\Delta w'_{ij}$也是上一次的修改量,β 在 0 和 1 之间取值。

4.4 算法的实现

4.2.1 的"网络的拓扑结构"一段曾经提到,要弄清楚 BP 网络,只需要考察二级网就可以了。而且,在绝大多数应用中选用的都是二级 BP 网络。因此,本节以典型的二级 BP 网为例,介绍它的实现。

设输入向量是 n 维的,输出向量是 m 维的,隐藏层有 H 个神经元,样本集含有 s 个样本。隐藏层和输出层的神经元的激活函数分别为 F_1、F_2。算法的主要数据结构如下:

$\boldsymbol{W}[H,m]$——输出层的权矩阵;

$\boldsymbol{V}[n,H]$——输入(隐藏)层的权矩阵;

$\Delta_o[m]$——输出层各联接权的修改量组成的向量;

$\Delta_h[H]$——隐藏层各联接权的修改量组成的向量;

$\boldsymbol{O}_1$——隐藏层的输出向量;

$\boldsymbol{O}_2$——输出层的输出向量;

$(\boldsymbol{X},\boldsymbol{Y})$——一个样本。

算法的主要实现步骤如下:

1 用不同的小伪随机数初始化 $\boldsymbol{W}$、$\boldsymbol{V}$;

2 初始化精度控制参数 ε、学习率 α;

3 循环控制参数 $E = \varepsilon + 1$;循环最大次数 M;循环次数控制参数 $N = 0$;

4 while $E>\varepsilon$ & $N<M$ do

  4.1 $N = N + 1$;$E = 0$;

  4.2 对每一个样本$(\boldsymbol{X},\boldsymbol{Y})$,执行如下操作:

    4.2.1 计算:$\boldsymbol{O}_1 = F_1(\boldsymbol{XV})$;$\boldsymbol{O}_2 = F_2(\boldsymbol{O}_1\boldsymbol{W})$;

    4.2.2 计算输出层的权修改量:for $i = 1$ to m

      4.2.2.1 $\Delta_o[i] = (1-\boldsymbol{O}_2[i])(\boldsymbol{Y}[i]-\boldsymbol{O}_2[i])$;

    4.2.3 计算输出误差:for $i = 1$ to m

      4.2.3.1 $E = E + (\boldsymbol{Y}[i] - O_2[i])^2$;

    4.2.4 计算隐藏层的权修改量:for $i = 1$ to H

4.2.4.1 $Z = 0$

4.2.4.2 for $j = 1$ to m

$Z = Z + W[i,j] \times \Delta_o[j]$;

4.2.4.3 $\Delta_h[i] = Z$;

4.2.5 修改输出层权矩阵:for $k = 1$ to H & $i = 1$ to m

4.2.5.1 $\boldsymbol{W}[k,i] = \boldsymbol{W}[k,i] + \alpha O_1[k]\Delta_o[i]$;

4.2.6 修改隐藏层权矩阵:for $k = 1$ to n & $i = 1$ to H

4.2.6.1 $\boldsymbol{V}[k,i] = \boldsymbol{V}[k,i] + \alpha O_1[k]\Delta_h[i]$;

建议读者在做实验时,可以将隐含层的神经元的个数 H 作为一个输入参数,实验一下,对同一个问题(相同的样本集),看在隐藏层中用多少个神经元能够得到最好的效果。同样,也可以同时将 ε、循环最大次数 M 等作为算法的输入参数。另一个建议是,在网络的调试阶段,在最外层循环内加一层控制,让系统在每循环若干次后,将误差测度、权矩阵输出,以便使调试者可以了解到训练的实际进程,也可在训练不收敛时及时地停止算法,以尽早地进行调整。这里所说的调整主要是指对权矩阵 **W**、**V** 的初值的调整。因为不同的初值可能会导致网络陷入局部极小点,而一旦陷入了局部极小点,网络就很难达到系统的精度要求。

另外,上述是对算法 4-1 的实现,读者可以对此实现进行适当的修改来实现算法 4-2。

4.5 算法的理论基础

BP 算法有很强的理论基础。算法对网络的训练被看成是在一个高维空间中寻找一个多元函数的极小点。事实上,我们不妨设网络含有 M 层,各层的联接矩阵分别为:

$$\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \cdots, \boldsymbol{W}^{(M)} \qquad 4-14$$

如果第 h 层的神经元有 H_h 个,则网络被看成一个含有

$$n \times H_1 + H_1 \times H_2 + H_2 \times H_3 + \cdots + H_M \times m \qquad 4-15$$

个自变量的系统。该系统将针对样本集:

$$\boldsymbol{S} = \{(\boldsymbol{X}_1, \boldsymbol{Y}_1), (\boldsymbol{X}_2, \boldsymbol{Y}_2), \cdots, (\boldsymbol{X}_s, \boldsymbol{Y}_s)\}$$

进行训练。我们取网络的误差测度为该网络相对于样本集中所有样本的误差测度的总和:

$$E = \sum_{p=1}^{s} E^{(p)} \qquad 4-16$$

式中 $E^{(p)}$ 为网络关于样本$(\boldsymbol{X}_p, \boldsymbol{Y}_p)$的误差测度。由上式可知,如果对任意的

$$(\boldsymbol{X}_p, \boldsymbol{Y}_p) \in \boldsymbol{S}$$

我们均能使 $E^{(p)}$ 最小,则就可使 E 最小。因此,为了后面的叙述简洁,我们用 w_{ij} 代表 $w_{ij}^{(h)}$,用 net_j 表示相应的神经元 AN_j 的网络输入,用 E 代表 $E^{(p)}$,用$(\boldsymbol{X}, \boldsymbol{Y})$代表$(\boldsymbol{X}_p, \boldsymbol{Y}_p)$,其中

$$\boldsymbol{X} = (x_1, x_2, \cdots, x_n)$$

$$Y = (y_1, y_2, \cdots, y_m)$$

该样本对应的实际输出为：

$$O = (o_1, o_2, \cdots, o_m)$$

我们用理想输出与实际输出的方差作为相应的误差测度：

$$E = \frac{1}{2}\sum_{k=1}^{m}(y_k - o_k)^2 \qquad 4-17$$

按照最速下降法，要求 E 的极小点，应该有：

$$\Delta w_{ij} \propto -\frac{\partial E}{\partial w_{ij}} \qquad 4-18$$

这是因为，$\frac{\partial E}{\partial w_{ij}}$为 E 关于 w_{ij} 的增长率，为了使误差减小，所以取 Δw_{ij} 与它的负值成正比。图 4－5 为相应的示意图。

图 4－5(a) 中，当$\frac{\partial E}{\partial w_{ij}}>0$ 时，系统当前所处的位置在极小点的右侧，所以，w_{ij} 的值应该减小，故此时 $\Delta w_{ij}<0$ 成立。图 4－5(b) 表示相反的情况，此时$\frac{\partial E}{\partial w_{ij}}<0$，系统当前所处的位置在极小点的左侧，所以 w_{ij} 的值应该增大，故此时 $\Delta w_{ij}>0$ 成立。

注意到式 4－17，需要变换出 E 相对于该式中网络此刻的实际输出的关系，因此，

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}} \qquad 4-19$$

而其中的

$$net_j = \sum_k w_{kj} o_k$$

所以

$$\frac{\partial net_j}{\partial w_{ij}} = \frac{\partial\left(\sum_k w_{kj} o_k\right)}{\partial w_{ij}} = o_i \qquad 4-20$$

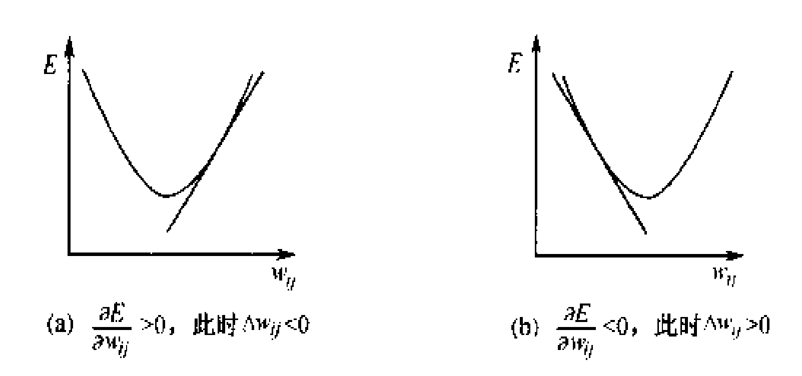


图 4－5　Δw_{ij} 与的关系示意图

将式 4－20 代入式 4－19，可以得到：

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}}$$

$$= \frac{\partial E}{\partial net_j} \cdot \frac{\partial\left(\sum_k w_{kj} o_k\right)}{\partial w_{ij}}$$

$$= \frac{\partial E}{\partial net_j} \cdot o_i$$

令

$$\delta_j = -\frac{\partial E}{\partial net_j} \qquad 4-21$$

根据式 4－18，可以得到：

$$\Delta w_{ij} = \alpha \delta_j o_i \qquad 4-22$$

其中，α 为比例系数，在这里为学习率。

下面的问题是求 4－21。显然，当 AN_j 是网络输出层的神经元时，net_j 与 E 的函数关系比较直接，从而相应的计算比较简单。但是，当 AN_j 是隐藏层的神经元时，net_j 与 E 的函数关系就不是直接的关系，相应的计算就比较复杂了。所以，需要按照 AN_j 是输出层的神经元和隐藏层的神经元分别进行处理。

1. AN_j 为输出层神经元

当 AN_j 为输出层神经元时，注意到：

$$o_j = f(net_j)$$

容易得到

$$\frac{\partial o_j}{\partial net_j} = f'(net_j) \qquad 4-23$$

从而

$$\delta_j = -\frac{\partial E}{\partial net_j}$$

$$= -\frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j}$$

$$= -\frac{\partial E}{\partial o_j} \cdot f'(net_j)$$

在注意到式 4－17

$$\frac{\partial E}{\partial o_j} = \frac{\partial\left(\frac{1}{2}\sum_{k=1}^{m}(y_k - o_k)^2\right)}{\partial o_j}$$

$$= \frac{1}{2}\frac{\partial(y_j - o_j)^2}{\partial o_j}$$

$$= -(y_j - o_j)$$

所以

$$\delta_j = (y_j - o_j) f'(net_j) \qquad 4-24$$

故，当 AN_j 为输出层的神经元时，它对应的联接权 w_{ij} 应该按照下列公式进行调整：

$$\begin{aligned} w_{ij} &= w_{ij} + \alpha\delta_j o_i \\ &= w_{ij} + \alpha f'(net_j)(y_j - o_j) o_i \end{aligned} \qquad 4-25$$

2. AN_j 为隐藏层神经元

当 AN_j 为隐藏层神经元的时候，式 4－21 中的 net_j 及其对应的 $o_j(= f(net_j))$ 在 E 中是不直接出现的，所以，这个偏导数不能直接求，必须进行适当的变换。由于 net_j 是隐藏层的，而式中 E 含的是输出层的神经元的输出，所以考虑将“信号”向网络的输出方向“推进”一步，使之与 $o_j = f(net_j)$ 相关：

$$\begin{aligned} \delta_j &= \frac{\partial E}{\partial net_j} \\ &= \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \end{aligned}$$

由于，$o_j = f(net_j)$，所以

$$\frac{\partial o_j}{\partial net_j} = f'(net_j)$$

从而有

$$\delta_j = -\frac{\partial E}{\partial o_j} \cdot f'(net_j) \qquad 4-26$$

注意到式 4－17

$$E = \frac{1}{2}\sum_{k=1}^{m}(y_k - o_k)^2$$

中的 o_k 是它的所有前导层的所有神经元的输出 o_j 的函数。当前的 o_j 通过它的直接后继层的各个神经元的输出去影响下一层各个神经元的输出，最终影响到式 4－17 中的 o_k。而目前只用考虑将 o_j 送到它的直接后继层的各个神经元。不妨假定当前层（神经元 AN_j 所在的层）的后继层为第 h 层，该层各个神经元 AN_k 的网络输入为

$$net_k = \sum_{i=1}^{H_h} w_{ik} o_i \qquad 4-27$$

所以，E 对 o_j 的偏导可以转换成如下形式：

$$\frac{\partial E}{\partial o_j} = \sum_{i=1}^{H_h}\left(\frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial o_j}\right) \qquad 4-28$$

再由式 4－27，可得

$$\frac{\partial net_k}{\partial o_j} = \frac{\partial\left(\sum_{i=1}^{H_h} w_{ik} o_i\right)}{\partial o_j} = w_{jk} \qquad 4-29$$

将式 4 - 29 代入式 4 - 28，可得

$$\frac{\partial E}{\partial o_j} = \sum_{i=1}^{H_h}\left(\frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial o_j}\right)$$

$$= \sum_{i=1}^{H_h}\left(\frac{\partial E}{\partial net_k} \cdot w_{jk}\right) \qquad 4-30$$

与式 4 - 21 中的 net_j 相比，式 4 - 30 中的 net_k 为较后一层的神经元的网络输入。所以，按照我们遵从的 Δw_{ij} 的计算是从输出层开始，逐层向输入层推进的顺序，当要计算 AN_j 所在层的联接权的修改量时，神经元 AN_k 所在层的 δ_k 已经被计算出来。而 $\delta_k = -\frac{\partial E}{\partial net_k}$，即式 4 - 30 中的 $\frac{\partial E}{\partial net_k}$ 就是 $-\delta_k$。从而

$$\frac{\partial E}{\partial o_j} = -\sum_{k=1}^{H_h}\delta_k w_{jk} \qquad 4-31$$

将其代入式 4 - 26，可得

$$\delta_j = \frac{\partial E}{\partial o_j} \cdot f'(net_j)$$

$$= -\left(-\sum_{k=1}^{H_k}\delta_k w_{jk}\right) \cdot f'(net_j)$$

即

$$\delta_j = \left(\sum_{k=1}^{H_k}\delta_k w_{jk}\right) \cdot f'(net_j) \qquad 4-32$$

由式 4 - 22

$$\Delta w_{ij} = \alpha\left(\sum_{k=1}^{H_k}\delta_k w_{jk}\right) \cdot f'(net_j)o_i$$

故，对隐藏层的神经元的联接权 w_{ij}，有

$$w_{ij} = w_{ij} + \alpha\left(\sum_{k=1}^{H_k}\delta_k w_{jk}\right) \cdot f'(net_j)o_i \qquad 4-33$$

4.6 几个问题的讨论

前面曾经提到过，BP 网络是应用最为广泛的网络。例如，它曾经被用于文字识别、模式分类、文字到声音的转换、图像压缩、决策支持等。但是，有许多问题困扰着该算法。尤其是如下五个问题，对 BP 网络有非常大的影响，有的甚至是非常严重的。下面对这五个问题进行简单的讨论。

1. 收敛速度问题

BP 算法最大的弱点是它的训练很难掌握，所以在上节我们特别建议读者在网络的调试阶段

加强对网络的监视。该算法的训练速度是非常慢的，尤其是当网络的训练达到一定的程度后，其收敛速度可能会下降到令人难以忍受的地步。作者曾经作过一个试验，对一个输入向量的维数为4，输出向量的维数为3，隐藏层有7个神经元的BP网络，算法在外层循环执行到5 000次之前，收敛速度较快。大约每迭代100次，误差可以下降0.001左右，但从第10 000次到第20 000次迭代，总的误差下降量还不到0.001。更严重的是，训练有时是发散的。

2. 局部极小点问题

从4.5节知道，BP算法用的是最速下降法，从理论上看，其训练是沿着误差曲面的斜面向下逼近的。对一个复杂的网络来说，其误差曲面是一个高维空间中的曲面，它是非常复杂不规则的，其中分布着许多局部极小点。在网络的训练过程中，一旦陷入了这样的局部极小点，用目前的算法是很难逃离出来的。所以，在上节的算法实现中曾提醒读者，要严密地监视训练过程，一旦发现网络在还未达到精度要求，而其训练难以取得进展时，就应该终止训练。因为此时网络已经陷入了一个局部极小点。在这种情况下，可以想办法使它逃离该局部极小点或者避开此局部极小点。“避开”的方法之一是修改 $\boldsymbol{W}$、$\boldsymbol{V}$ 的初值，重新对网络进行训练。因为开始“下降”位置的不同，会使得网络有可能避开该极小点。但是，由于高维空间中的曲面是非常复杂的，所以，当网络真的可以“躲开”该局部极小点时，它还有可能陷入其他的局部极小点。因此，一般来讲，对局部极小点采用“躲开”的办法并不是总有效的。较好的方法是当网络掉进局部极小点时，能使它逃离该局部极小点，而向全局极小点继续前进。后面将介绍的统计方法在一定的程度上可以实现这一功能。但是，统计方法会使网络的训练速度变得更慢。

Wasserman在1986年提出，将Cauchy训练与BP算法结合起来，可以在保证训练速度不被降低的情况下找到全局极小点。

3. 网络瘫痪问题

在训练中，权可能变得很大，这会使神经元的网络输入变得很大，从而又使得其激活函数的导函数在此点上的取值很小。根据式4-5到4-11，此时的训练步长会变得非常小，进而导致训练速度降得非常低，最终导致网络停止收敛。这种现象叫做网络瘫痪。因此，我们强调，在对网络的联接权矩阵进行初始化时，要用不同的小伪随机数。

4. 稳定性问题

前面曾经提到，BP算法必须将整个训练集一次提交给网络，再对它进行联接权的调整，而且最好使用算法4-2，用整个样本集中各样本所要求的修改量的综合实施权的修改。这种做法虽然增加了一些额外的存储要求，但却能获得较好的收敛效果。

显然，如果网络遇到的是一个连续变化的环境，它将变成无效的。由此看来，BP网络难以模拟生物系统。BP网络缺乏的这种可塑性将在第8章介绍的ART模型中得到解决。

5. 步长问题

从上节可以看出，BP网络的收敛是基于无穷小的权修改量，而这个无穷小的权修改量预示着需要无穷的训练时间。这显然是不行的。因此，必须适当地控制权修改量的大小。

显然，如果步长太小，收敛就非常慢，如果步长太大，可能会导致网络的瘫痪和不稳定。较好的解决办法是设计一个自适应步长，使得权修改量能随着网络的训练而不断变化。一般来说，在

训练的初期,权修改量可以大一些,到了训练的后期,权修改量可以小一些。1988 年,Wasserman 曾经提出过一个自适应步长算法,该算法可以在训练的过程中自动地调整步长。

练 习 题

1. 叙述 BP 算法的基本思想。

2. BP 算法对人工神经网络技术的发展起到了很大的作用,它得到了非常广泛的研究和应用。请你分析这种网络模型能被广泛应用的主要原因。

3. 对 BP 网络的神经元的激活函数有什么特殊要求?为什么?

4. 在直观上如何解释 BP 网络在训练期间的误差估计阶段各层的误差估计?

5. 分析算法 4-1 的第 4 步,求该步被执行一次所需完成的乘法和加法各多少次。

6. 分析算法 4-2,指出该算法是如何消除样本顺序对结果的影响的。你认为,如果一个 BP 网络在完成训练后,又遇到一个新的样本,是否可以直接用这个样本对 BP 网络进行"附加训练"?请解释你给出的答案。

7. 请自行选择一个实例实现算法 4-2。

8. 对你在 4-7 中实现的算法,调整隐藏层神经元的个数,观察隐藏层神经元的个数对网络训练的收敛速度和网络的计算精度的影响。

9. 修改算法 4-2,使它能够按照式 4-12 调整 w_{ij}。

10. 简述困扰 BP 算法的几个问题。

11. 你是否能找到一种解决困扰 BP 网络的局部极小点的方法?

12. 对 BP 网络的性能进行分析,指出它将适应哪些方面的应用。

第五章 对 传 网

BP 网作为第一个性能较好的多级网络被 PDP 小组的研究者们在 1986 年重新提出并引起世人的广泛关注后，Robert Hecht - Nielson 在 1987 年提出了对传网（Counterpropagation Networks，简记为 CPN）。与 BP 网相比，CPN 的训练速度要快很多，所需的时间大约是 BP 网所需时间的 1%。但是，它的应用面却因网络的性能而比较窄。

从网络的拓扑结构来看，CPN 与 BP 网类似，CPN 是一个两层的神经网络，只不过这两层执行的训练算法是不同的。所以，CPN 是一个异构网。与同构网相比，网络的异构性使它更接近于人脑。因为研究表明，在人脑中确实存在有各种特殊的模块，它们用来完成不同的运算。例如，在听觉通道的每一层，其神经元与神经纤维在结构上的排列与频率的关系十分密切，对某一些频率，其中某些相应的神经元会获得最大的响应。这种听觉通道上的神经元的有序排列一直延续到听觉皮层。尽管许多低层次上的神经元是预先排列好的，但高层次上的神经元的组织则是通过学习自组织形成的。

在 Robert Hecht - Nielson 提出的 CPN 中，神经元被分布于两层，它们分别执行较早些时候出现的两个著名算法：Kohonen 1981 年提出的自组织映射（Self - organization map，简记为 SOM）和 Grossberg 1969 年提出的散射星（Outstar）。人们将执行自组织映射的层称为 Kohonen 层，执行散射星算法的层则被称为 Grossberg 层。按这种方法将这两种算法组合在一起后所获得的网络，不仅提供了一种设计多级网训练算法的思路，解决了多级网络的训练问题，突破了单级网的限制，而且还使得网络具有了许多新的特点。如前所述，多级网络的训练问题主要是在解决隐藏层神经元相应的联接权调整时，需要通过隐藏层神经元的理想输出来实现相关误差的估计。然而，它们对应的理想输出又是未知的。我们知道，在无导师训练中是不需要知道理想输出的，因此可以考虑让网络的隐藏层执行无导师学习。这是解决多级网络训练的另一个思路。实际上，CPN 就是将无导师训练算法与有导师训练算法结合在一起，用无导师训练解决网络隐藏层的理想输出未知的问题，用有导师训练解决输出层按系统的要求给出指定的输出结果的问题。

Kohonen 提出的自组织映射由四部分组成，包括一个神经元阵列（用它构成 CPN 的 Kohonen 层），一种比较选择机制，一个局部互联，一个自适应过程。实际上，这一层将实现对输入进行分类的功能。所以，该层可以执行无导师的学习，以完成对样本集中所含的分类信息的提取。

Grossberg 层主要用来实现类表示。由于相应的类应该是用户所要求的，所以，对应每一个输入向量，用户明确地知道它对应的理想输出向量，故，该层将执行有导师的训练。

两层的有机结合，就构成了一个映射系统。所以，有人将 CPN 看成一个有能力进行一定的推广的查找表（Look - up table）。它的训练过程就是将输入向量与相应的输出向量对应起来。

这些向量可以是二值的，也可以是连续的。一旦网络完成了训练，对一个给定的输入就可以给出一个对应的输出。网络的推广能力表明，当网络遇到一个不太完整的、或者是不完全正确的输入时，只要该“噪音”是在有限的范围内，CPN 都可以产生一个正确的输出。这是因为 Kohonen 层可以找到这个含有噪音的输入应该属于的分类，而对应的 Grossberg 层则可以给出该分类的表示。从整个网络来看，就表现出一种推广能力。这使得网络在模式识别、模式完善、信号加强等方面可以有较好的应用。

另外，上述映射的逆映射如果存在的话，可以通过对此网的简单扩展，来实现相应的逆变换。这被称为全对传网。

本章将介绍 CPN 的网络结构，Kohonen 层与 Grossberg 层的正常运行，对传网关于输入向量的预处理要求与处理方法，Kohonen 层的训练算法及其权矩阵的初始化方法；Grossberg 层的训练；完整的对传网。

5.1 网络结构

按照对传网所表达出来的意思，它应该是可以完成信号的双向变换和传递的。不过，为了叙述简洁、易懂起见，我们先介绍 CPN 的向前传递阶段，并简称它为单向 CPN，而完整的 CPN(双向网)在原理上是一样的，将留在最后介绍。同时，为了区分起见，在这里简称完整的 CPN 为全对传网，并统一地将它们记为 CPN。

图 5－1 给出了简化的单向 CPN 的拓扑结构。从表面上看，它和前面介绍的二级网络是相同的，但是在运行过程中，它们实现的策略却是不同的。因此，对一个网络，除了它的拓扑结构外，其运行机制也是确定网络结构(如：同构、异构)和性能的重要因素。

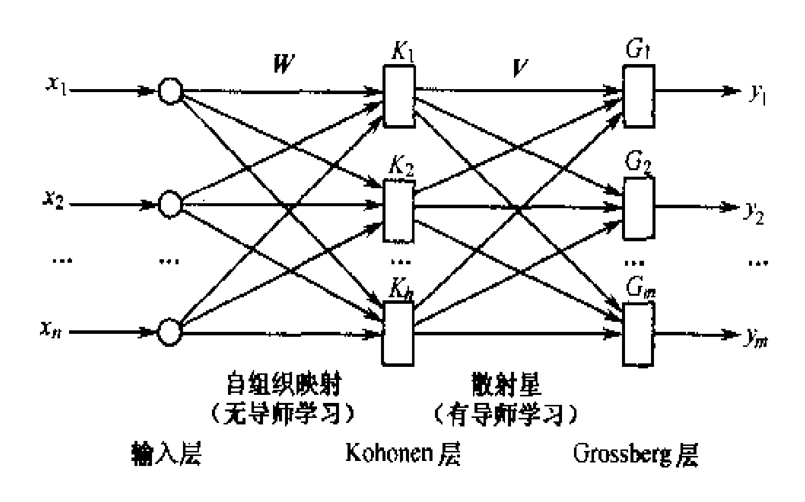


图 5－1　单向 CPN 结构

第 0 层(输入层)的神经元只起到将输入向量广播到 Kohonen 层的作用，使得输入向量的每一个分量可以被按照一定的联接权传递到第 1 层(Kohonen 层)的每个神经元。Kohonen 层的联接权矩阵用 W 表示：按照惯例，w_{ij}表示输入向量的第 i 个分量到 Kohonen 层的第 j 个神经元的

联接权。第 2 层叫做 Grossberg 层，Kohonen 层的第 i 个神经元到 Grossberg 层的第 j 个神经元的联接权用 v_{ij} 表示，它们放在一起构成 Grossberg 层的权矩阵。从该网络来看，将网络的联接(Connection)层的个数用来计算网络的层数是方便的。在这里，仍然用向量：

$$X = (x_1, x_2, \cdots, x_n)$$

$$Y = (y_1, y_2, \cdots, y_m)$$

分别表示输入向量和输出向量。其中，n 为输入向量的维数，m 为输出向量的维数，同时也是 Grossberg 层神经元的个数。h 为 Kohonen 层神经元的个数。

另外，由于该网络的特殊性，将以 Kohonen 层的神经元为"中心"来讨论一些问题。为了叙述方便，使用如下记号：

$$W_1 = (w_{11}, w_{21}, \cdots, w_{n1})$$

$$W_2 = (w_{12}, w_{22}, \cdots, w_{n2})$$

$$\cdots\cdots$$

$$W_h = (w_{1h}, w_{2h}, \cdots, w_{nh})$$

$$V_1 = (v_{11}, v_{12}, \cdots, v_{1m})$$

$$V_2 = (v_{21}, v_{22}, \cdots, v_{2m})$$

$$\cdots\cdots$$

$$V_h = (v_{h1}, v_{h2}, \cdots, v_{hm})$$

与其他网络一样，CPN 有两种工作模式：训练模式和正常工作模式。在训练模式下，对一个样本(X, Y)，Kohonen 层按照 X 的要求进行无导师学习(自组织映射要求的学习)。对应地，Grossberg 层则按照理想输出 Y 的要求调整相应的联接权。在正常工作模式下，给定的输入向量 X 被加在网络上，通过 Kohonen 层的自组织映射而使该层的神经元处于相应的状态(在最简单的情况下，该层的神经元将仅有一个处于激发态)，由这个状态表达的信息被 Grossberg 层转换成用户要求的形式输出。下面分别对它们进行讨论。

5.2 网络的正常运行

在 CPN 中，Kohonen 层和 Grossberg 层不仅执行不同的训练算法，而且它们在正常运行阶段所执行的算法也是不同的。

5.2.1 Kohonen 层

在最简单的情况下，Kohonen 层是以"强者占先、弱者退出"(the winner takes all)的方式工作的。这就是说，对一个给定的输入向量，Kohonen 层的神经元处于激烈的竞争中，最后，谁获得的网络输入最大，谁就处于激发态，而其他的就处于抑制态。同时规定，处于激发态的神经元输出为 1，而处于抑制态的神经元输出为 0。在本书中，不再讨论多个神经元可以同时处于激发状态的复杂情况的处理。实际上，复杂情况的处理是类似的，只是在处理细节上有一定的差别。有关

这类问题，留给读者去思考。

根据网络的特点，讨论与 Kohonen 层的每个神经元相关联的权向量 $\boldsymbol{W}_1, \boldsymbol{W}_2, \cdots, \boldsymbol{W}_h$。对 Kohonen 层的每个神经元 $K_j(1 \leqslant j \leqslant n)$，它所获得的网络输入：

$$\begin{aligned} knet_j &= \boldsymbol{XW}_j \\ &= (x_1, x_2, \cdots, x_n)(w_{1j}, w_{2j}, \cdots, w_{nj})^{\mathrm{T}} \\ &= w_{1j}x_1 + w_{2j}x_2 + \cdots + w_{nj}x_n \end{aligned} \quad 5-1$$

$$\boldsymbol{KNET} = (knet_1, knet_2, \cdots, knet_h)$$

$K_1, K_2, \cdots, K_h$ 的输出 $k_1, k_2, \cdots, k_h$ 构成向量：

$$\boldsymbol{K} = (k_1, k_2, \cdots, k_h) \quad 5-2$$

其中，对 $1 \leqslant j \leqslant h$

$$k_j = \begin{cases} 1 & knet_j = \max\{knet_1, knet_2, \cdots, knet_h\} \\ 0 & \text{其他} \end{cases} \quad 5-3$$

从式 5-1 和 5-3 可知，对每个输入向量 $\boldsymbol{X}$，Kohonen 层的每个神经元 $K_j(1 \leqslant j \leqslant n)$所获得的网络输入就是输入向量 $\boldsymbol{X}$ 与它相对应的联接权向量 $\boldsymbol{W}_j$ 的点积。由此，可以给出 Kohonen 层的输出这样的几何解释：Kohonen 层的每个神经元 $K_j(1 \leqslant j \leqslant n)$对应一个分类，同时，它对应的联接权向量 $\boldsymbol{W}_j$ 与使神经元 K_j 激发的这一组 $\boldsymbol{X}$ 的“几何距离”最近，所以 $\boldsymbol{W}_j$ 就是这一组 $\boldsymbol{X}$ 的代表。请读者注意，Kohonen 层输出的几何解释指出了 Kohonen 层训练所要求的方式。

5.2.2 Grossberg 层

按照人工神经网络的运行模型的定义，对 Grossberg 层的每个神经元 $G_j(1 \leqslant j \leqslant m)$，它所获得的网络输入为

$$\begin{aligned} gnet_j &= \boldsymbol{K}(v_{1j}, v_{2j}, \cdots, v_{hj})^{\mathrm{T}} \\ &= (k_1, k_2, \cdots, k_h)(v_{1j}, v_{2j}, \cdots, v_{hj})^{T} \\ &= k_1 v_{1j} + k_2 v_{2j} + \cdots + k_h v_{hj} \end{aligned}$$

如上所述，假定 Kohonen 层按照最简单的方式工作。这样，对应每一个输入向量，该层只有唯一的一个神经元处于激发态而输出 1，其他的神经元均输出 0。假设这个唯一输出 1 的神经元为 K_o，由式 5-4，此时，Grossberg 层的神经元 $G_j(1 \leqslant j \leqslant m)$所获得的网络输入为

$$\begin{aligned} gnet_j &= k_1 v_{1j} + k_2 v_{2j} + \cdots + k_h v_{hj} \\ &= v_{oj} \end{aligned} \quad 5-5$$

此时

$$\begin{aligned} \boldsymbol{GNET} &= (gnet_1, gnet_2, \cdots, gnet_m) \\ &= (v_{o1}, v_{o2}, \cdots, v_{om}) \end{aligned}$$

该向量正好就是与 K_o 对应的向量 $\boldsymbol{V}_o$。如果此时 Grossberg 层的神经元的激活函数为恒等映射函数，则 Grossberg 层的输出（也就是该 CPN 的输出），就是 $\boldsymbol{V}_o$。可以认为，$\boldsymbol{V}_o$ 为 K_o 对应的 $\boldsymbol{W}_o$ 的变换形式。由于 $\boldsymbol{V}_o$ 的各个分量都是从 K_o 到 Grossberg 层各神经元的联接权，所以此算法被

称为散射星(Outstar)。

因此,如果将 CPN 用于模式的完善,则此时有

$$n = m$$

网络接受含有噪音的输入模式$(x_1, x_2, \cdots, x_n)$,而输出去掉噪音后的模式为$(v_{o1}, v_{o2}, \cdots, v_{om})$。

综上所述,对一个输入向量 $\boldsymbol{X}$,CPN 首先检查它与 $\boldsymbol{W}_1, \boldsymbol{W}_2, \cdots, \boldsymbol{W}_h$ 中的哪一个最为接近。最接近的 $\boldsymbol{W}_j$ 对应的神经元K_j 被激发。一方面,这说明 CPN 将样本空间中的输入向量分成了 h 类,而且它们的代表分别是 $\boldsymbol{W}_1, \boldsymbol{W}_2, \cdots, \boldsymbol{W}_h$。所以,当面临的问题比较复杂,需要将输入向量分更多的类的时候,可以适当地增加 Kohonen 层神经元的个数。另一方面,CPN 的分类实际上是在寻找与输入向量 $\boldsymbol{X}$ 最接近的 $\boldsymbol{W}_o$。因此,它们的初始化是非常重要的。这不仅会影响到训练的速度,而且还会直接地影响到网络的精度。在 Kohonen 层完成模式的分类之后,对 Grossberg 层而言,它就是以适当的方式输出被激发的 K_o 所对应的向量 $\boldsymbol{V}_o$。因此,$\boldsymbol{V}_o$ 实际上就是 $\boldsymbol{W}_o$ 的一个变换。所以,在训练过程中,对 $\boldsymbol{W}_1, \boldsymbol{W}_2, \cdots, \boldsymbol{W}_h$,将努力使它们代表各类 $\boldsymbol{X}$ 的共同特征。例如,可以将这一类 $\boldsymbol{X}$ 的平均值作为相应的 $\boldsymbol{W}_o$ 的训练目标;对 $\boldsymbol{V}_1, \boldsymbol{V}_2, \cdots, \boldsymbol{V}_h$,则是让它们对应地去表现 $\boldsymbol{W}_1, \boldsymbol{W}_2, \cdots, \boldsymbol{W}_h$ 所代表的样本集中的$\boldsymbol{X}$ 对应的理想输出 $\boldsymbol{Y}$ 的共同特征。

5.3 Kohonen 层的训练

上节已介绍过,Kohonen 层按照输入向量 $\boldsymbol{X}$ 的相似度完成对它们的分类。分类的结果由 Grossberg 层转换成要求的输出形式。Kohonen 层使用自组织算法,按照无导师方式学习;因此,对一个给定的输入向量,Kohonen 层的哪一个神经元被激发,人们事先是不知道的,算法只能保证将不相似的向量区分开来,而使相似的向量能激发 Kohonen 层的同一个神经元。为了较好地实现这一思想,算法强烈希望将输入向量、以及 Kohonen 层各神经元对应的 Kohonen 层的权向量 $\boldsymbol{W}_1, \boldsymbol{W}_2, \cdots, \boldsymbol{W}_h$ 进行规范化处理,使它们均为单位向量,以保证竞争的"公平性"。

5.3.1 输入向量的预处理

输入向量的预处理就是对输入向量进行单位化处理。要完成的主要是比较简单的数学运算,只要将向量的每一个分量除以该向量的模就可以了。设

$$\boldsymbol{X} = (x_1, x_2, \cdots, x_n)$$

$\boldsymbol{X}$ 的模:

$$\|\boldsymbol{X}\| = \mathrm{sqrt}((x_1, x_2, \cdots, x_n)(x_1, x_2, \cdots, x_n)^{\mathrm{T}})$$

$\boldsymbol{X}$ 的单位化向量:

$$\begin{aligned}\boldsymbol{X}' &= (x'_1, x'_2, \cdots, x'_n) \\ &= (x_1/\|\boldsymbol{X}\|, x_2/\|\boldsymbol{X}\|, \cdots, x_n/\|\boldsymbol{X}\|)\end{aligned}$$

即,对 $1 \leqslant i \leqslant n$,有

$$x'_j = x_j/\|\boldsymbol{X}\|$$

为了叙述的方便起见，在本章中，后面出现的输入向量均被假定是单位向量，除非给予特殊说明。

5.3.2 训练

为了使 $\boldsymbol{W}_1, \boldsymbol{W}_2, \cdots, \boldsymbol{W}_h$ 能较好地代表网络所给出的样本集中的输入向量的类划分，对一个给定的输入向量 $\boldsymbol{X}$，首先必须在 $\boldsymbol{W}_1, \boldsymbol{W}_2, \cdots, \boldsymbol{W}_h$ 中找出目前谁最接近 $\boldsymbol{X}$。Kohonen 层中的神经元的网络输入为 $\boldsymbol{X}$ 与它对应的权向量的点积，而点积最大者（$\boldsymbol{W}_o$）拥有与 $\boldsymbol{X}$ 最大的相似度。所以，应该用它来代表 $\boldsymbol{X}$ 所在的类。而为了使它更好地代表 $\boldsymbol{X}$，需进一步调整该 $\boldsymbol{W}_o$，使它变得更接近 $\boldsymbol{X}$，以便更好地代表 $\boldsymbol{X}$。但是，这种调整幅度不能太大，必须是有限度的，否则，算法将会出现强烈的抖动。据此，有下列算法：

算法 5-1　Kohonen 层训练算法

1　对所有的输入向量，进行单位化处理；

2　对每个样本（$\boldsymbol{X}, \boldsymbol{Y}$）执行下列过程

　2.1　for j = 1 to h do

　　2.1.1　根据式 5-1 计算 $knet_j$；

　2.2　求出最大的 $knet_o$：

　　2.2.1　$max = knet_1; o = 1$;

　　2.2.2　for j = 1 to h do

　　　2.2.2.1　if $knet_j > max$ then $\{max = knet_j; o = j\}$;

　2.3　计算 $\boldsymbol{K}$：

　　2.3.1　for j = 1 to h do $k_j = 0$;

　　2.3.2　$k_o = 1$;

　2.4　使 $\boldsymbol{W}_o$ 更接近 $\boldsymbol{X}$：

　　$\boldsymbol{W}_o^{(new)} = \boldsymbol{W}_o^{(old)} + \alpha(\boldsymbol{X} - \boldsymbol{W}_o^{(old)})$;

　2.5　对 $\boldsymbol{W}_o^{(new)}$ 进行单位化处理。

在算法的 2.1 步中，未给出 $knet_j$ 的具体计算方法。它的计算除了包括输入向量的各个分量的加权和之外，还有一部分为通过侧联接而获得的来自本层的神经元的输入。按照本节目前所设定的最简单的工作模式，这些来自侧联接的信号对神经元来说均应该是抑制性的。它们用来实现本层神经元之间的竞争。在程序的实现中，可以用一个单独的 h 行、h 列的二维数组实现。除对角线元素取正值外，其他元素都应该取负值。同时读者还应该注意，侧联接的权值要与输入层的联接的权值在大小上有一个协调，使它们都能较好地发挥作用。

算法的步骤 2.4 使用公式

$$\boldsymbol{W}_o^{(new)} = \boldsymbol{W}_o^{(old)} + \alpha(\boldsymbol{X} - \boldsymbol{W}_o^{(old)}) \qquad 5-8$$

来使 $\boldsymbol{W}_o^{(new)}$ 变得比 $\boldsymbol{W}_o^{(old)}$ 更接近 $\boldsymbol{X}$。其中，$\boldsymbol{W}_o^{(new)}$ 表示 $\boldsymbol{W}_o$ 在本次被调整后的值，$\boldsymbol{W}_o^{(old)}$ 表示

W_o 在本次调整之前的值。α 为学习率：

$$\alpha \in (0,1) \tag{5-9}$$

可以证明，使用式 5－8 对 $W_o^{(new)}$ 进行变换后，确实会使得 $W_o^{(new)}$ 变得比 $W_o^{(old)}$ 更接近 X。事实上：

$$\begin{aligned} W_o^{(new)} &= W_o^{(old)} + \alpha(X - W_o^{(old)}) \\ &= W_o^{(old)} + \alpha X - \alpha W_o^{(old)} \\ X - W_o^{(new)} &= X - [W_o^{(old)} + \alpha(X - W_o^{(old)})] \\ &= X - W_o^{(old)} - \alpha X + \alpha W_o^{(old)} \\ &= X(1-\alpha) - W_o^{(old)}(1-\alpha) \\ &= (1-\alpha)(X - W_o^{(old)}) \end{aligned}$$

从而

$$\begin{aligned} \frac{X - W_o^{(new)}}{X - W_o^{(old)}} &= \frac{(1-\alpha)(X - W_o^{(old)})}{X - W_o^{(old)}} \\ &= 1 - \alpha \end{aligned}$$

由式 5－9 知 $(1-\alpha)<1$。所以，$W_o^{(new)}$ 比 $W_o^{(old)}$ 更接近 X。

图 5－2 为式 5－8 的几何表示，从图中可以看出，$W_o^{(new)}$ 确实比 $W_o^{(old)}$ 更接近 X。图中的圆为单位圆。所以，从原点到该圆上任一点的向量都是单位向量。

算法的步骤 2.5 要求对 $W_o^{(new)}$ 进行规范化处理，这是为了保证后续的训练仍然能以相同的方式进行。当然，包括对输入向量 X 的单位化处理在内，都要求算法付出新的代价。但是，对 X、W 进行了单位化处理之后，会使本层的训练更有效地进行。因此，为 X、W 的单位化所付出的代价是值得的。在用程序实现的时候，建议读者设计出一个专门的子程序来完成此项工作。

在图 5－2 中，被进行过单位化处理的 $W_o^{(new)}$ 将在未被单位化处理的 $W_o^{(old)}$ 的基础上，继续延长到单位圆上，也就是说，它的方向不变，只是长度延长。

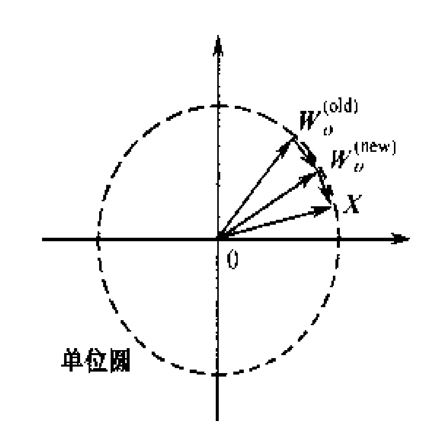


图 5－2　式 5－8 的几何意义

从上述讨论不难看出，算法的第一步要求对 W_j 进行初始化处理是很有必要的。

学习率 α 在0和1之间取值。实践经验告诉我们，在训练的初期，α 一般取0.7左右，它将随着训练的进展不断变小。这是因为，算法的运行表明，到了后期，W_j 已经对应一组 X，而不是一个。所以，在这个阶段，不能允许一个 X 对 W_j 的影响太大。因为，此时如果再让 X 对 W_j 的影响过大，它会过多地破坏 W_j 对该类中其他输入向量的表达，等这些向量在下次迭代中重新出现时，它又会对 W_j 进行较大的调整，这样一来，训练就会陷入抖动。当然，在训练的初期，由于各神经元正处于待选状态，所以当遇到一个合适的输入向量时，就尽量地去表示它。如果在开始训练时取 α 为1，就相当于让 W_j 直接取 X，当样本顺序比较合适，分布也比较恰当时，这种做法是比较好的。但是，在大多数情况下，它会导致 W_j 进入一个不太合适的"类"的表示中。也会造成分类的"人为"畸形，使得有的类非常小，有的类又太大，甚至导致有的 X 被放入错误的类中。因此，除非情况明显，建议读者一般不要取 α 为1。

实际上，如果在训练之前能用其他适当的方法事先给问题一个粗略的分类，并从这个分类中提取一个较有代表性的向量构成样本集，这时，Kohonen层的每个 W_j（对应一个神经元）的值直接对应一个 X，则可取 α 为1，使得获胜者的权向量直接取 X。这种做法又启发我们采用训练和直接设定权向量的方式来完成该层的训练。即在进行初始化时，直接用初选的样本集中的样本来对 W 进行初始化，然后再适当地扩大样本集，用扩大后的样本集实施上述算法，完成对Kohonen层的训练。

一般来说，一个类含有许多向量。这个类对应的 W_j 应该是样本集中这一类向量（输入向量部分）的平均值。

5.4 Kohonen层联接权的初始化方法

联接权的初始化，是人工神经网络训练的第一步。通常使用一些不同的小伪随机数作为它们的初值。按照上几节的分析，Kohonen层的联接权对应于该层的每个神经元而构成的向量应该是单位向量。这是为后面的训练所作的准备。

一个非常重要的问题是，样本集中的输入向量决定着网络对其将进行的分类。按照上节的分析，在理想的情况下，$W_1, W_2, \cdots, W_h$ 的初值应该依照样本集中的输入向量的分布来确定。而实际上，一般很难弄清楚样本集中的输入向量的分布情况。实际上，Kohonen层的重要工作之一就是要找出样本集中输入向量的分布情况。当然，如果样本集中输入向量的分布是均匀的，就可以按照随机均匀分布的方式去完成对 $W_1, W_2, \cdots, W_h$ 的初始化工作。然而，在多数情况下，样本集中的输入向量的分布并不是均匀的。所以，用不同的小伪随机数作为联接权的初值的做法就不一定是适应的，这可能造成严重的不平衡。因为不同的小伪随机数会使这些权向量的初值均匀地分布在一个高维"球面"上，而对应的输入向量通常的不均匀性会使得它们趋向于在这个高维"球面"的某一部分聚集。随机的权向量的大多数会因为离输入向量"太远"而无法获得匹配，从而使相应的神经元输出总是为0，这样一来，该神经元和它对应的这组联接权以及它对应的Grossberg层的那一组联接权就被浪费掉了。与此同时，其中会有一部分神经元代表的类含有过多的输入向量，因此，又难以对这些向量进行人们所期望的分类。

按照上述分析,最为理想的是能够根据输入向量的实际分布来设置 Kohonen 层各神经元对应的权向量。实际上,要做到这一点是很困难的。虽然在上面曾经提到过可以根据实际问题进行一定的“预处理”,但是这样一来不仅增加了困难,而且这一困难部分是在牺牲了 Kohonen 层的自动分类功能后出现的。所以,一般来说,不会采用此方法,尤其是对复杂问题来说更是如此。

从上述分析知道,寻求一些有效的权矩阵初始化方法是非常有必要的。在这里介绍 Kohonen 层联接权的几种初始化方法。它们各有特点,可以参考使用。

1. 凸状组合法

与使用一些不同的小伪随机数对联接权进行初始化不同,凸状组合法是用同一个数来初始化 Kohonen 层的每一个联接权。

仍然设输入向量是 n 维的,输出向量为 m 维的,Kohonen 层有 h 个神经元,则对任意的 $i(1 \leqslant i \leqslant n)$、$j(1 \leqslant j \leqslant h)$,取

$$w_{ij} = \frac{1}{\mathrm{sqrt}(n)} \quad 5-10$$

其中,$\mathrm{sqrt}(n)$表示 n 的算术平方根。

与联接权的这一初值相对应,要求对输入向量进行下列变换,以便能较好地适应之。设

$$\boldsymbol{X} = (x_1, x_2, \cdots, x_n)$$

则变换后的 $\boldsymbol{X}$(记为 $\boldsymbol{X}'$)为

$$\boldsymbol{X}' = (x'_1, x'_2, \cdots, x'_n)$$

其中

$$\boldsymbol{X}'_j = \lambda x_j + \frac{1-\lambda}{\sqrt{n}} \quad 5-11$$

在训练的初期阶段,λ 的值非常小,此时

$$\begin{aligned} \boldsymbol{X}'_j &= \lambda x_j + \frac{1-\lambda}{\sqrt{n}} \\ &= \frac{1}{\sqrt{n}} + \lambda\left(x_j - \frac{1}{\sqrt{n}}\right) \\ &\approx \frac{1}{\sqrt{n}} \end{aligned}$$

这使得

$$\boldsymbol{X}' \approx \left(\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \cdots, \frac{1}{\sqrt{n}}\right)$$

成立,这也使得 $\boldsymbol{X}'$成为近似的单位向量,而且和 Kohonen 层的神经元对应的权向量近似。

随着训练的进行,λ 趋近于 1,从而使 $\boldsymbol{X}'$趋近于$\boldsymbol{X}$,进而使得 $\boldsymbol{W}_j$ 趋近于一组$\boldsymbol{X}$ 的平均值。

这种方法较好地解决了初始权向量的分布与输入向量分布难以一致的问题。但是,由于在训练中,$\boldsymbol{X}'$是逐渐趋近于$\boldsymbol{X}$ 的,相应地,也就有一组 $\boldsymbol{X}'$的平均值逐渐地“收敛”于对应的这一组 $\boldsymbol{X}$ 的平均值。这说明相应的训练算法能够使 $\boldsymbol{W}_j$ 不断地调整自己的“运动方向”,以追踪运动中

的目标。显然,让算法跟踪一个运动的目标要比让它逼近一个固定目标困难得多。因此,该方法虽然能够工作,但是,它的收敛速度将是非常慢的。

2. 添加噪音法

凸状组合法实际上是通过在输入向量中加噪音而使输入向量重新分布,只不过这种噪音较一致。如果按照通常的习惯,使用一些不同的小伪随机数来初始化权向量,得到一个均匀分布的权向量,然后,在输入向量中添加随机的噪声,使它们也变得是均匀分布的,从而解决两者分布不同的矛盾。这种方法叫做添加噪音法。

显然,在这种方法中,在输入向量中加进适当的随机噪音后,使得输入向量的分布变成均匀的,网络中的所有权向量就可以比较容易地被这些均匀分布的"输入向量"所"捕获"。然后,算法逐渐地去掉所加的噪音,最终,权向量就会变得按输入向量的分布而分布了。

与凸状组合法类似,添加噪音法也是让 W_j 不断地调整自己的"运动方向",去追踪不断变化的目标。试验表明,这种方法的收敛速度比凸状组合法更慢。

3. 初期全调法

Kohonen 层的初始权向量的分布与输入向量分布的不一致,会造成一些权向量获得过多的输入匹配,而同时又有一些权向量得到很少的匹配,有的甚至得不到匹配。凸状组合法、添加噪音法通过给输入向量加噪音来避免这一现象发生。初期全调法则是直接从避免这种现象入手,去解决问题。该方法在训练的初期,对应一个输入向量,允许多个神经元同时处于激发状态。这样,相应地就有多个权向量获得调整。被激发神经元的多少可以通过一个限定值来控制。这个限定值可以是同时可能被激发的神经元的最大个数,也可以是网络输入超过某一阈值的所有的神经元。随着算法的进展,逐渐减少被激发的神经元的最大个数或者逐渐提高所定的阈值,最后达到对一个输入向量只有一个神经元激发。

这种方法的另一种实现是,对每一个输入向量,虽然每次只有一个神经元处于激发态,但是,在训练的初期,算法不仅调整"获胜"的神经元对应的权向量,而且对其他的权向量也作适当的调整。随着训练的推进,被调整的将逐渐变成是与"获胜"的神经元对应的权向量"最近"的一些权向量,这样,被调整的范围逐渐缩小,直到最终只有"获胜"的神经元对应的权向量才被调整。

在上述第二种实现中,调整的范围也需要有一个适当的度量。而且,这个度量应该是随时间变化且容易计算的。另外,除了"获胜"的神经元对应的权向量外,其他的权向量的"适当调整"也是需要认真考虑的问题。显然,这个"适当的调整量"应该比"获胜"者对应的权向量的调整量要小。而且,随着训练算法的进行,这一调整量也应逐渐减少。

4. DeSieno 法

该方法是 D. DeSieno 1988 年提出的。他根据 Kohonen 层的初始权向量的分布与输入向量的分布不一致可能造成有的神经元会获得过多的匹配的现象,直接限制 Kohonen 层的每个神经元对应的权向量可以匹配的输入向量的最大个数。当某一个权向量所获得的匹配向量超过给定的数后,它的阈值就被临时提高,以增加其他神经元"获胜"的机会。一般地,这个最大个数可以是样本总数的 $1/h$。

这种方法的问题是当最应该被某个神经元对应的权向量匹配的输入向量在较后的时候被输

入时，它可能被拒绝，从而造成网络精度的损失，应该引起读者的注意。

通过对 Kohonen 层的训练，可以使它抽象出输入向量集中的一些统计特性。1988 年，Kohonen 曾经证明，在一个被完全训练过的网中，随机选取的输入向量与任何给定的权向量是最接近的概率是 $1/h$。这就是说，就一般的统计特性看来，每个按均匀分布初值的权向量都具有相同的被匹配的概率。

5.5 Grossberg 层的训练

相对于 Kohonen 层的训练而言，Grossberg 层的训练要容易许多。在这里，算法将依据 Kohonen 层被调整的权向量对应的神经元，来调整它所对应的 Grossberg 层的权向量。所以，与前面介绍的 Kohonen 层训练算法的一些调整相对应，这里介绍的 Grossberg 层的训练也应该随之有相应的调整。

设$(\boldsymbol{X},\boldsymbol{Y})$为一样本，当 $\boldsymbol{X}$ 被输入到 CPN 中后，Kohonen 层的神经元 K_o 被激发，此时 K_o 的输出 k_o 为 1，其他的神经元输出为 0。用下式对 $\boldsymbol{V}_o$ 进行调整：

$$v_{oj} = v_{oj} + \alpha(y_j - v_{oj}) \quad (1 \leqslant j \leqslant m) \qquad 5-12$$

其中 v_{oj}为 Kohonen 层的神经元 K_o 到 Grossberg 层的神经元 G_j 的联接权，α 为学习率。写成向量形式，有

$$\boldsymbol{V}_o^{(\text{new})} = \boldsymbol{V}_o^{(\text{old})} + \alpha(\boldsymbol{Y} - \boldsymbol{V}_o^{(\text{old})}) \qquad 5-13$$

为了方便，将式 5-8 重新写在下面：

$$\boldsymbol{W}_o^{(\text{new})} = \boldsymbol{W}_o^{(\text{old})} + \alpha(\boldsymbol{X} - \boldsymbol{W}_o^{(\text{old})})$$

比较这两个式子，不难看出，Grossberg 层的训练与 Kohonen 层的训练是类似的：与 kohonen 层的训练是将 $\boldsymbol{W}_o$ 训练成为这一类 $\boldsymbol{X}$ 的平均值相对应，Grossberg 层的训练则是将对应的 $\boldsymbol{X}_o$ 训练成为这一类 $\boldsymbol{X}$ 对应的 $\boldsymbol{Y}$ 的平均值。

由此可见，Kohonen 层的训练可以说是以 $\boldsymbol{X}$ 为目标，而 Grossberg 层的训练则是以 $\boldsymbol{Y}$ 为目标的。这就是说，从单级网的角度来看，这两层的训练都是有导师训练。但是，对整个网络来说，$\boldsymbol{X}$ 是输入向量，$\boldsymbol{Y}$ 是输出向量。所以，又可以认为前者是无导师训练，后者是有导师训练。

上面我们分别叙述了 CPN 的 Kohonen 层的训练和 Grossberg 层的训练。但实际上，CPN 这两层的训练既可以同时进行，也可以分开进行。当分开进行时，网络训练的实现算法可以被设计得效率要高一些，但它们执行的原理算法都是一样的。

设 CPN 的样本集：

$$\boldsymbol{S} = \{(\boldsymbol{X},\boldsymbol{Y}) \mid \boldsymbol{X},\boldsymbol{Y}\ \text{是一个样本对，其中}\ \boldsymbol{Y}\ \text{是}\ \boldsymbol{X}\ \text{对应的理想输出向量}\}$$

算法 5-2 将对两层同时进行训练。

算法 5-2　CPN 训练算法一

0　对 $\boldsymbol{W}$、$\boldsymbol{V}$ 进行初始化；

1　对所有的输入向量，进行单位化处理；

2 对每个样本($\boldsymbol{X}$, $\boldsymbol{Y}$)执行下列过程:

2.1 for $j = 1$ to h do

2.1.1 根据式 5-1 计算 $knet_j$;

2.2 求出最大的 $knet_o$:

2.2.1 $max = knet_1$; $o = 1$;

2.2.2 for $j = 1$ to h do

2.2.2.1 if $knet_j > max$ then $\{max = knet_j; o = j\}$;

2.3 计算 $\boldsymbol{K}$:

2.3.1 for $j = 1$ to h do $k_j = 0$;

2.3.2 $k_o = 1$;

2.4 使 $\boldsymbol{W}_o$ 更接近 $\boldsymbol{X}$:

$\boldsymbol{W}_o^{(new)} = \boldsymbol{W}_o^{(old)} + \alpha(\boldsymbol{X} - \boldsymbol{W}_o^{(old)})$;

2.5 对 $\boldsymbol{W}_o^{(new)}$ 进行单位化处理;

2.6 使 $\boldsymbol{V}_o$ 更接近 $\boldsymbol{Y}$:

$\boldsymbol{V}_o^{(new)} = \boldsymbol{V}_o^{(old)} + \alpha(\boldsymbol{Y} - \boldsymbol{V}_o^{(old)})$。

实际上,上述算法只是在算法 5-1(Kohonen 层的训练算法)的基础上增加了对 $\boldsymbol{W}$、$\boldsymbol{V}$ 的初始化和步骤 2.6。下面算法的效率要高一些,但它要求有额外的存储。

算法 5-3 CPN 训练算法二

0 对 $\boldsymbol{W}$、$\boldsymbol{V}$ 进行初始化;

0′ 清空 Kohonen 层各神经元对应的纪录表:

for $j = 1$ to h do $\boldsymbol{SK}_j = \Phi$;

1 对所有的输入向量,进行单位化处理;

2 对每个样本($\boldsymbol{X}_s$, $\boldsymbol{Y}_s$)执行下列过程:

2.1 for $j = 1$ to h do

2.1.1 根据式 5-1 计算 $knet_j$;

2.2 求出最大的 $knet_o$:

2.2.1 $max = knet_1$; $o = 1$;

2.2.2 for $j = 1$ to h do

2.2.2.1 if $knet_j > max$ then $\{max = knet_j; o = j\}$;

2.3 计算 $\boldsymbol{K}$:

2.3.1 for $j = 1$ to h do $k_j = 0$;

2.3.2 $k_o = 1$;

2.4 使 $\boldsymbol{W}_o$ 更接近 $\boldsymbol{X}_s$:

$$W_o^{(new)} = W_o^{(old)} + \alpha(X_s - W_o^{(old)});$$

2.5　对 $W_o^{(new)}$ 进行单位化处理；

2.6　将 Y_s 放入 SK_o：

$$SK_o = SK_o \cup \{Y_s\};$$

3　for $j = 1$ to h do

$V_j =$ SK_j 中各向量的平均值。

与算法 5－2 相比，算法 5－3 首先是增加了额外的存储 $SK_1, SK_2, \cdots, SK_h$。它们都是集合变量。算法 5－3 直接将 Grossberg 层的迭代式训练改成一次计算型的，效率上有所提高。

读者可以进一步考虑该算法的优化等问题。例如，可以将算法 5－3 中的集合变量 SK_1，$SK_2, \cdots, SK_h$ 改为其他存储量更小而且更容易实现的变量。另一个值得注意的问题是关于算法 5－3 的 2.6 步，在 X_s 激发 K_o 时，Y_s 被放入到 SK_o 中，这会隐藏一个更为严重的错误。如果样本(X_s, Y_s)在某一次循环中激发的是神经元 K_o，此时 X_s 将被放入到 SK_o 中，而如果因为 K_o 对应的权向量在后续的训练中进行了较大的改动，则当 X_s 被再次输入时，可能会激发另一个神经元 K_i，这里 $i \neq o$。按照算法 5－3，此时 Y_s 除了被放入 SK_i 外，它还曾经被放入 SK_o 中，而此时 X_s 对 K_o 对应的 W_o 的影响可能已被"淡化"。因此，这要求在将一个理想输出向量放入某个 SK 时做更多的工作。

5.6 补充说明

前几节，对 CPN 基本结构和基本算法进行了介绍，为了叙述方便起见，叙述中忽略了一些内容。为了使内容较为完善，本节对上述讨论进行适当的扩展。

1. 全对传网

首先，将图 5－1 给出的简单单向 CPN 扩展为可以实现对传的双向 CPN，图 5－3 是全对传网的结构示意图。它在经过训练后，可以实现信号的双向传送，以实现相应的变换和逆变换。

该网的训练继续使用前面讨论的 CPN 的训练算法。在这里，对训练样本(X, Y)，将其中的 X 和 Y 并置在一起，构成一个向量，暂时用 XY 表示。所以，所给的向量既是对传网的输入向量，又是该输入向量对应的理想输出向量。由此意义来看，在最理想的情况下，该网络将实现一个恒等映射。从图 5－3 给出的拓扑结构也可以看出这一点：只要将 X'、Y'调换一下位置就可以了。

在实际运行中，误差总是存在的。所以，在 XY 中，用 X 来训练网络，使它在接受到 X 时，可以输出 X'，而用 Y 来训练网络，使它在接受到 Y 时，可以输出 Y'。

训练中，XY 被从输入端输入网络，经过 Kohonen 层的识别，被激发的神经元对应的 Kohonen 层的权向量依据 XY 进行调整，相应地，该神经元对应的 Grossberg 层的权向量依据 YX 进行调整。

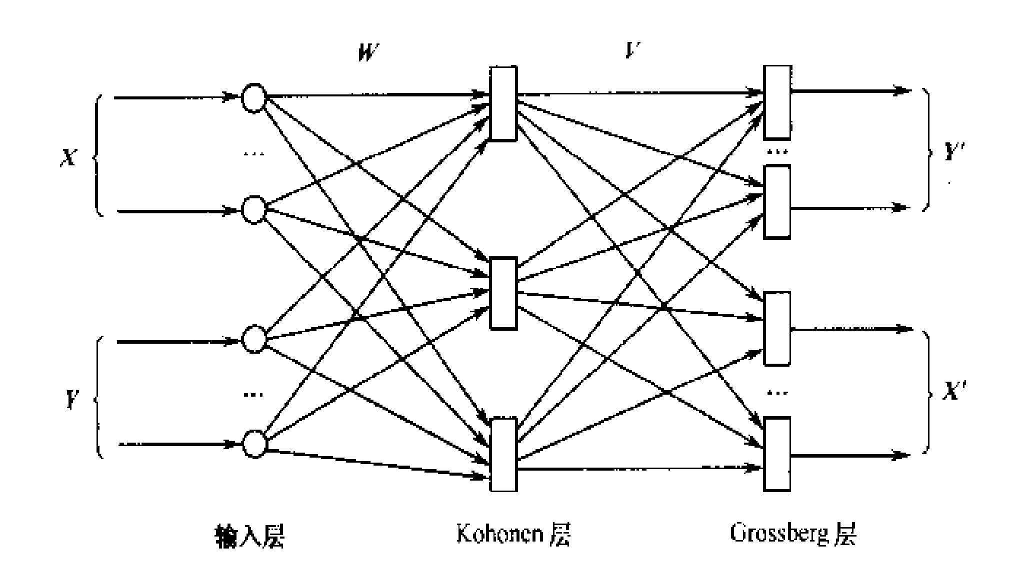


图 5-3 全对传网

在网络完成学习后，投入正常运行时，给网络输入向量 **XY**，网络将输出 **Y′X′**；如果只输入 **X**，而将 **Y** 置为 **0**，这时网络输出的 **Y′X′** 中的 **Y′** 就是 **X** 的映射结果。相应地，如果只输入 **Y**，而将 **X** 置为 **0**，这时网络输出 **Y′X′** 中的 **X′** 就是 **Y** 的映射结果。

按照上述分析，对传网用于数据的压缩和相应的解压等所需要进行变换和逆变换的处理应该是比较适应的。另外，由于网络的 Kohonen 层的分类功能和 Grossberg 层的变换功能，该网络可以用于问题的分类表达、决策支持、规划等方面。

2．非简单工作方式问题

前面的讨论都是基于这样一个假设：对一个输入向量，Kohonen 层的神经元有且仅有一个神经元处于激发态。这一限制虽然使讨论变得比较简单，但它较大地限制了网络的功能。而且当掌握了 CPN 的基本结构和基本算法后，略作修改，就可以去掉这一限制。

对给定的任意输入向量，CPN 的 Kohonen 层的各神经元可以给出不同的输出，然后，训练算法可以将此输出作为对应的神经元所对应的 Kohonen 层、Grossberg 层的权向量的修改因子：输出值较大的，表明该输入向量与该神经元对应的类较接近。所以，它对应的权向量的修改量就大；输出值较小的，表明该输入向量与该神经元对应的类较远。所以，它对应的权向量的修改量就小。当 Kohonen 层的神经元的激活函数为阈值函数时，神经元的输出只能是 0 或 1，此时所有输出为 1 的神经元对应的 Kohonen 层、Grossberg 层的权向量被修改，而所有输出为 0 的神经元对应的 Kohonen 层、Grossberg 层的权向量保持不变。对应一个输入，最多只能有几个神经元处于激发态，要由问题本身去决定。

在非简单工作方式下，还有一些细节问题。例如，由于分类的需要，将被要求将非零输出的神经元的输出看成一个向量，并对其进行单位化处理。相应的算法也会有一些细节上的调整。所有这些都留给读者去思考，不再详细叙述。当对应一个输入向量，网络中可以处于激发态的神经元的个数为 1 时，网络就退化成最简单的形式。

练 习 题

1. 在拓扑结构上，对传网与二级 BP 网络是相同的，而实际上它们的网络结构却不同，请叙述它们之间的差别。

2. 叙述自组织映射(SOM)层执行的学习策略和工作策略。

3. 叙述散射星(Outstar)层执行的学习策略和工作策略。

4. 请指出自组织映射(SOM)层和散射星(Outstar)层是如何有机地结合起来构成对传网，不仅解决了多层网络的训练算法问题，而且还使对传网的功能大大地超过了原来的单级网络。

5. 既然自组织映射(SOM)层和散射星(Outstar)层分别执行不同的算法，是否可以把它们从"物理"上分开，让它们独立运行，而将自组织映射(SOM)的运算结果"传给"散射星(Outstar)，让它对"数据"做进一步的处理，从而取得与对传网相同或者相似的效果？为什么？

6. 叙述对传网训练中向量的预处理作用和方法。

7. 分析对传网中散射星(Outstar)所构成的 Kohonen 层联接权的初始化方法，并对这些方法进行适当的比较。

8. 比较算法 5－2 和算法 5－3，指出它们在性能上的差异。

9. 请较详细地设计出算法 5－3 的实现。

10. 对传网是否会面临局部极小点问题？为什么？

11. 受自组织映射(SOM)和散射星(Outstar)相结合构成了一个新的人工神经网络模型的启发，是否可以将其他的不同模型或者用更多的模型串接(或者联合)起来，构成一种新的网络，以获得新的系统性能？

12. 改造算法 5－3，使它能够满足网络按非简单方式工作的需要。

第六章　非确定方法

前三章介绍的网络所用训练算法有一个共同的特点:对应一个样本的一次迭代,算法都是根据相应样本、神经元的状态甚至联接权的当前值和联接权的上一次的调整值等具体的值计算出各个联接权的调整量。也就是说,算法所执行的计算是确定的。但是,人脑所含的神经元却是按照概率工作的。在人脑中,一个神经元在某一时刻是处于激发状态(兴奋状态)还是处于抑制状态是具有一定的随机性的。既然人工神经网络是用来模拟人脑工作的,那么,它除了可以按照前几章讨论的确定的方法来计算联接权的调整量外,也应该允许神经元"按照概率工作",使其状态具有一定的随机性。

为了将本章讲授的方法与前面讲授的方法相区别,将前面讲授的方法叫做确定的方法。相对地,本章将讲授的方法叫做非确定的方法,又称为统计方法(Statistical Method)。

事实上,人们研究非确定方法的出发点除了因为生物神经系统是按照概率原理进行工作的外,还希望采用统计方法能获得更好的效果,例如使网络逃离局部极小点。

在人工神经网络中,按照非确定的方法,与联接权的直接计算的确定方法相对应,它还应该可以是根据一定的概率实现随机调整的;与神经元的状态完全由其所获得的网络输入和它的激活函数来决定相对应,其状态也可以按照概率原理来工作,使它的激发和抑制具有一定的随机性。基于这一点,我们说,非确定的方法在人工神经网络的训练和正常运行等两种运行模式下均可以实现。

本章将介绍统计网络的基本训练算法、模拟退火算法与收敛分析、Cauchy 训练、人工热与临界温度在训练中的使用、BP 算法与 Cauchy 训练的结合。

6.1　基本的非确定训练算法

基本的非确定训练算法的基本思想是:从所给的网络中"随机地选取一个联接权",对该联接权提出一个"伪随机调整量",当用此调整量对所选的联接权进行修改后,如果"被认为""这个修改改进了网络的性能",则保留此调整;否则放弃本次调整。"被认为"在这里表示除了接受那些直接改善网络的性能的调整外,还应按照一定的概率接受那些使网络的性能暂时变坏,但在总趋势上却是在改善网络的性能的调整。例如,图 6-1 中所示的系统从 A 点向右移动时,在到达 D 点之前所表现出的情况就是这样。

下面我们考虑用有导师方式对一个多级网络进行训练。设网络的样本集为:

$$\boldsymbol{S} = \{(\boldsymbol{X}_1,\ \boldsymbol{Y}_1),(\boldsymbol{X}_2,\ \boldsymbol{Y}_2),\cdots,(\boldsymbol{X}_s,\ \boldsymbol{Y}_s)\}$$

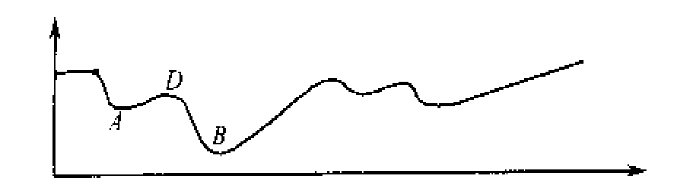


图 6-1　局部极小点示意图

其中：

$$X = (x_1, x_2, \cdots, x_n)$$

$$Y = (y_1, y_2, \cdots, y_m)$$

分别为输入向量和对应的理想输出向量。网络有 M 层，从第 1 层到第 M 层的联接矩阵依次为

$$\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \cdots, \boldsymbol{W}^{(M)}$$

它的拓扑结构如图 6-2 所示。

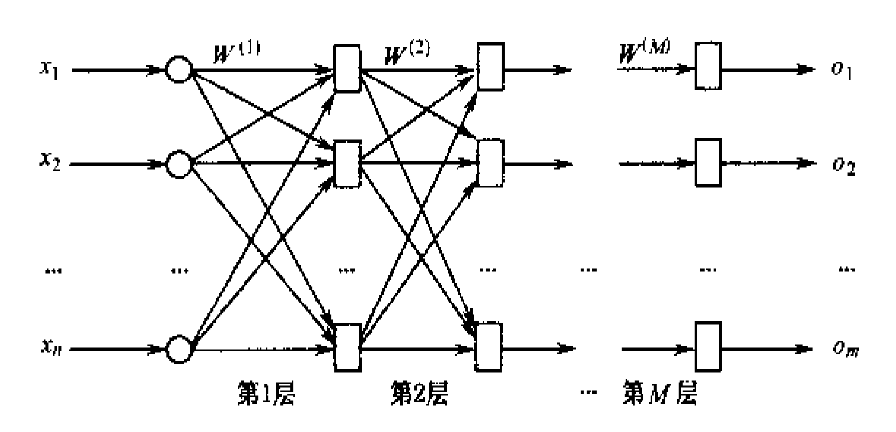


图 6-2　多级前馈网

算法 6-1　基本统计训练算法

1　从样本集 $\boldsymbol{S}$ 中取一样本 $(\boldsymbol{X}, \boldsymbol{Y})$；

2　将 $\boldsymbol{X}$ 输入到网络中，计算出实际输出 $\boldsymbol{O}$；

3　求出网络关于 $\boldsymbol{Y}$、$\boldsymbol{O}$ 的误差测度 E；

4　随机地从 $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \cdots, \boldsymbol{W}^{(M)}$ 中选择一个联接权 $w_{ij}^{(p)}$；

5　生成一个小随机数 $\Delta w_{ij}^{(p)}$；

6　用 $\Delta w_{ij}^{(p)}$ 修改 $w_{ij}^{(p)}$；

7　用修改后的 $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \cdots, \boldsymbol{W}^{(M)}$ 重新计算 $\boldsymbol{X}$ 对应的 $\boldsymbol{O}'$；

8　求出网络关于 $\boldsymbol{Y}$、$\boldsymbol{O}'$ 的误差测度 E'；

9　如果 $E' < E$，则保留本次对 $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \cdots, \boldsymbol{W}^{(M)}$ 的修改，否则，根据概率判断本次修改是否有用，如果认为有用，则保留本次对 $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \cdots, \boldsymbol{W}^{(M)}$ 的修改，如果认为本次修改无用，则放弃它；

10　重复上述过程，直到网络满足要求。

训练的目标是，对样本集中的所有样本，网络能获得最小的误差测度 E。所以，网络的训练过程，就是极小化 E 的过程。因此，有时又将网络的误差测度函数叫做网络的目标函数（*Objective Function*）。一般地，采用实际输出与理想输出的方差之和作为网络的目标函数。

算法的第 4 步是从 $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \cdots, \boldsymbol{W}^{(M)}$ 中随机地选择一个联接权 w_{ij}，如果第 h 层的神经元有 H_h 个，此步则是从

$$n \times H_1 + H_1 \times H_2 + H_2 \times H_3 + \cdots + H_M \times m$$

个变量中随机地选一个。仅此一项，就有许多种选法。可见，本算法所含的计算量是很大的。

算法的第 5 步是要随机地产生 $w_{ij}^{(p)}$ 的修改量 $\Delta w_{ij}^{(p)}$。可以用系统中的伪随机数发生器来产生它，也可以根据网络当前的状态按照所谓的“能量”函数的分布去计算它。相比之下，后一种方法留给用户的计算量要大一些，但它的“估计精度”一般应该高一些，这对提高整个算法的效率是非常有好处的。将在下一节介绍有关方法。

算法第 9 步，当 $E' < E$ 不成立时，按照绝对的标准，表明本次修改使目标函数的值增加了。所以，就本次修改的局部而言，它与算法是对目标函数进行极小化是相背的。但是，此时算法并不是立即拒绝接受这个修改，它还要根据概率判断本次修改是否有用。其主要原因是，在网络向目标函数的“全局极小点”收敛的过程中，有时会掉进“局部极小点”。为了使网络从局部极小点中逃离出来，必须允许目标函数暂时被变坏。

算法的第 10 步是控制第 1 步到第 9 步被重复执行，“直到网络满足要求”。相关的问题是：

1. 以什么样的标准来判断网络是否满足要求？

2. 在用一个样本对网络的某一个联接权进行修改后（该修改可能是有效的，也可能是无效的），是随机地抽取另一个联接权进行重复，还是再选择下一个样本进行重复？

3. 对一个选定的样本，每次是否可以选取若干个联接权进行修改？如果可以，还应做什么工作？

这些问题留给读者去思考。在这里，先讨论严重影响着网络训练是否收敛的局部极小点问题。

在第 4 章介绍 BP 算法时，曾经提到过困扰网络训练的局部极小点问题。这个问题的难度在于，在对一个 M 级的多级网络的训练中，从数学的角度讲，面对的是求一个 $n \times H_1 + H_1 \times H_2 + H_2 \times H_3 + \cdots + H_M \times m$ 元函数的极小点的问题。这就是说，算法要在一个极其复杂的高维曲面上运行，并寻找它的全局极小点。BP 算法采用的是最速下降法，它一旦使网络掉进了局部极小点，就难以逃离出来。

图 6-1 为局部极小点的示意图。图中的小球代表目标函数，如果在训练过程中，目标函数落在局部极小点 A，当随机的权修改量较小时，目标函数难以脱离 A 点所在的凹区：因为目标函数在局部极小点 A 附近（不包括 A 点）所取得值大于它在 A 点的值。所以，当系统离开 A 点时，会使得目标函数有所上升。此时，若算法拒绝所有的使目标函数上升的联接权调整，网络的目标函数就只能停留在 A 点，永远无法达到全局极小点 B。其表现形式为，网络的精度离要求较远，但网络的目标函数却再也无法降低。

但是，如果权的修改量太大，目标函数可能会在 A、B 两点间来回跳动（即网络的抖动）而无法停在理想的极小点上。

一种较好的办法是：让算法从取较大的修改量开始，然后逐渐地减少这个修改量，使得网络在寻找全局极小点区域时，有足够的"能力"从局部极小点跳出来；一旦它进入到全局极小点区域，联接权的修改量将变得小到使网络没有足够的"能量"跳出来。因此，对搜寻全局极小点的问题，可以总结出如下思路：

- 联接权修改量
  - 太小：落到 A 点后很难逃离
  - 太大：导致在 A、B 两点来回抖动
- 解决办法：权修改量由大变小

这一点还提醒我们，联接权的修改量的大小应该是和网络的"能量"相关的。现在的问题是，如何来度量网络在某一时刻的"能量"？又如何利用这个"能量"？模拟退火算法就是"借用"金属热处理中的"退火"原理解决问题的。

6.2 模拟退火算法

按照前面的介绍，网络的目标函数为网络的误差测度。当误差测度较大时，表明网络离全局极小点比较远。因此，它应该具有较大的"能量"，以便使它可以逃离可能"路过"的局部极小点而不至于被限制在那里；当误差测度较小时，表明网络已进入"全局极小点区域"，因此它只需要较小的"能量"，以便使其下降到全局极小点，并且该"能量"又是不足以使它离开"全局极小点区域"的。因此，考虑使用网络的目标函数作为网络能量的测度。从而有图 6-3 所示的分析。

图 6-3 下半部分简单地给出了在金属热加工中金属的温度与它所含能量的关系。在金属热加工过程中，当金属的温度超过它的熔点(Melting Point)时，原子就会激烈地随机运动。与所有其他的物理系统类似，原子的这种运动趋向于寻找其能量的极小状态。在这个能量的变迁过程中，开始时，温度非常高，使得原子具有很高的能量。随着温度的不断降低，金属逐渐冷却，金属中原子的能量就越来越小，最后达到所有可能的最低点。这就是全局极小点。这种方法是通过升高温度来提高金属中原子的能量，使得这些原子有能力摆脱其原来的能量状态而最后达到一个更加稳定的状态－－全局极小能量状态。在金属的退火过程中，其能量的状态分布由如下关系确定：

$$P(E) \propto \exp\left(-\frac{E}{kT}\right) \qquad 6-1$$

其中，$P(E)$——系统处于具有能量 E 的状态的概率；

k——Boltzmann 常数；

T——系统的绝对温度(Kelvin)。

注意到

$$\lim_{T \to +\infty}\left(\exp\left(-\frac{E}{kT}\right)\right) = 1 \qquad 6-2$$

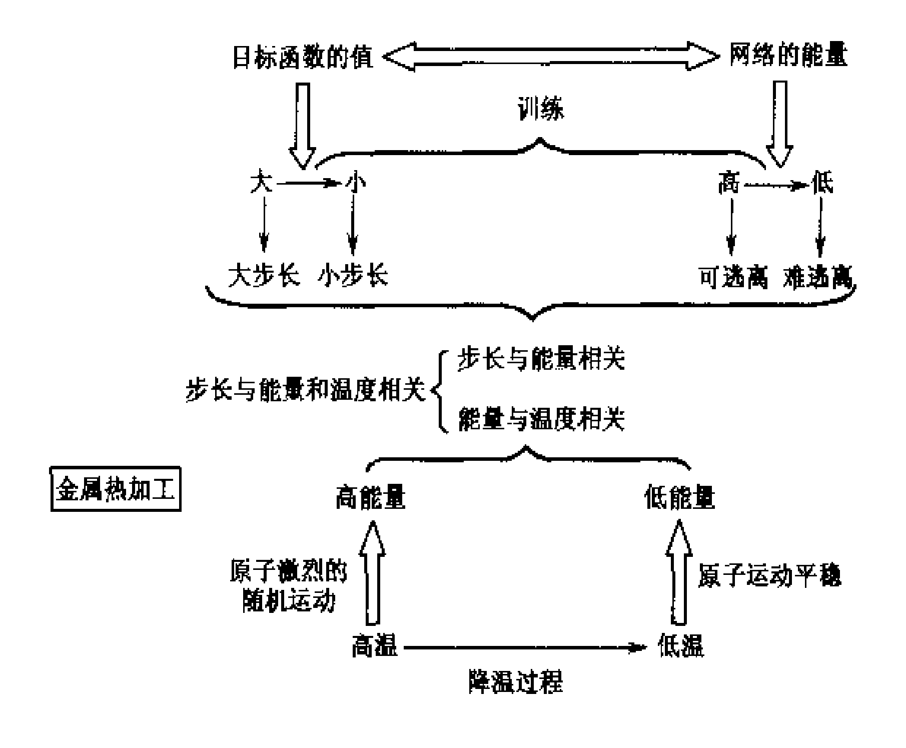


图 6-3　步长和能量、温度的关系

在温度足够高时，上式中的 T 起主要作用，从而对所有的能量状态 E，$P(E)$ 均接近于 1。这表明，系统处于高能量状态的概率与处于低能量状态的概率是一样的。但是随着温度的降低，E 在决定 $\exp\left(-\frac{E}{kT}\right)$ 的值时的作用将越来越明显：E 越大，$P(E)$ 就越小，即系统处于高能量状态的可能性就越小。显然，由该函数的连续性知，存在一个温度 T，使得系统的温度在不高于 T 时，它处于高能量状态的可能性小于处于低能量状态的可能性。当温度降到 0 时，系统就很难处于一个高能量的状态。事实上，根据式 6-1，有：

(1) 在高温情况下，T 足够大，对系统所能处的任意能量状态 E，随着 $\frac{E}{kT}$ 趋近于 0，$\exp\left(-\frac{E}{kT}\right)$ 将趋近于 1；

(2) 在中温情况下，T 比较小，在 $\frac{E}{kT}$ 中，E 的大小对 $P(E)$ 有较大的影响，设 $E_1>E_2$，则 $\frac{E_1}{kT}>\frac{E_2}{kT}$，使得 $-\frac{E_1}{kT}<-\frac{E_2}{kT}$。从而，$P(E_2)>P(E_1)$，即系统处于高能量状态的可能性小于处于低能量状态的可能性。

(3) 在低温情况下，T 非常小，设 $E_1>E_2$，

$$\lim_{T\to 0}\frac{P(E_1)}{P(E_2)}=\lim_{T\to 0}\left(\frac{\exp\left(-\frac{E_1}{kT}\right)}{\exp\left(-\frac{E_2}{kT}\right)}\right)$$

$$=\lim_{T\to 0}\left(\exp\left(-\left(\frac{E_1}{kT}-\frac{E_2}{kT}\right)\right)\right)$$

$$= \lim_{T\to 0}\left(\exp\left(-\frac{E_1 - E_2}{kT}\right)\right)$$

$$= \lim_{T\to 0}\left(\left(\frac{1}{\exp(E_1 - E_2)}\right)^{\frac{1}{kT}}\right)$$

式中，

$$E_1 - E_2 > 0$$

所以，

$$\exp(E_1 - E_2) > 1$$

从而，

$$\frac{1}{\exp(E_1 - E_2)} < 1$$

然而

$$\lim_{T\to 0}\frac{1}{kT} = \infty$$

所以，

$$\lim_{T\to 0}\frac{P(E_1)}{P(E_2)} = \lim_{T\to 0}\left(\left(\frac{1}{\exp(E_1 - E_2)}\right)^{\frac{1}{kT}}\right) = 0$$

即，

$$P(E_2) \gg P(E_1) \qquad 6-3$$

这就是说，当温度趋近于0时，系统处于低能量状态的概率远远大于它处于高能量状态的概率。实际上，由于系统处于较低能量状态的概率最大为1，上式表明，此时系统处于较高能量状态的概率几乎为0。因此，在此种情况下，系统几乎不可能处于高能量状态。

从上述分析知道，如果将网络的训练看成是让网络寻找最低能量状态的过程，取网络的目标函数为它的能量函数，再定义一个初值较大的数为人工温度 T。同时，在网络的这个训练过程中，依据网络的能量和温度来决定联接权的调整量（称为步长）。这种做法与金属的退火过程(Annealing)非常相似。所以，人们将这种方法叫做模拟退火组合优化法。

模拟退火组合优化法的基本思想是：随机地为系统选择一个初始状态$\{w_{ij}^{(p)}\}$，在此初始状态下，给系统一个小的随机扰动 $\Delta w_{ij}^{(p)}$，计算系统的能量变化：

$$\Delta E = E(\{w_{ij}^{(p)} + \Delta w_{ij}^{(p)}\}) - E(\{w_{ij}^{(p)}\}) \qquad 6-4$$

若

$$\Delta E < 0$$

则此扰动被接受；如果

$$\Delta E \geqslant 0$$

则此扰动依据概率

$$\exp\left(-\frac{\Delta E}{kT}\right) \qquad 6-5$$

判断是否被接受。如果此扰动被接受，则系统从状态 $\{w_{ij}^{(p)}\}$ 变换到状态 $\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\}$；否则，系统的状态保持不变。如此重复下去。在这个过程中，逐渐地降低温度 T。所得的系统状态序列 $\{w_{ij}^{(p)}\}$ 将满足

$$f = c(T)\exp\left(-\frac{E(\{w_{ij}^{(p)}\})}{kT}\right) \qquad 6-6$$

分布。其中

$$c(T) = \frac{1}{\sum\exp\left(-\frac{E(\{w_{ij}^{(p)}\})}{kT}\right)} \qquad 6-7$$

显然，当取 E 为网络的实际输出和理想输出之间的差时，由于网络的实际输出是它的

$$n \times H_1 \times H_2 \times \cdots \times H_p \times \cdots \times H_M \times m$$

个联接权 $w_{ij}^{(p)}$ 的函数，所以，系统的能量函数就是 $w_{ij}^{(p)}$ 的函数。

算法 6-2 模拟退火算法

1 初始化各层的联接权矩阵 W；定义人工温度 T 的初值；

2 对每一个温度 T 重复如下过程：

2.1 选取一个样本，计算其输出与目标函数 $E(\{w_{ij}^{(p)}\})$；

2.2 随机地从 $\{w_{ij}^{(p)}\}$ 中选取一个 $w_{ij}^{(p)}$；

2.3 按一定的算法产生 $w_{ij}^{(p)}$ 的一个调整量 $\Delta w_{ij}^{(p)}$；

2.4 按照 $\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\}$ 重新计算相应的输出和目标函数 $E(\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\})$；

2.5 $\Delta E = E(\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\}) - E(\{w_{ij}^{(p)}\})$；

2.6 if $\Delta E > 0$ then

2.6.1 按均匀分布在[0, 1]区间取一随机数 r；

2.6.2 按 Boltzmann 分布计算接受本次调整的概率：

$$P(E(\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\})) = \exp\left(-\frac{E(\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\})}{kT}\right);$$

2.6.3 if $P(E(\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\})) < r$ then 转 2.2；

2.7 用 $\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\}$ 代替 $\{w_{ij}^{(p)}\}$；

2.8 if 样本集中还有未被选用的样本 then 转 2.1；

3 判断在此温度下，检验 Metropolis 抽样是否稳定。如不稳定，则直接转 2；

4 降低温度 T；

5 如果 T 足够小，则结束，否则，转 2。

关于该算法，有如下几点说明：

1、算法的第 2 步原则上应该对每一个样本调整每一个权，调整的顺序是随机的；在叙述中，并未严格给出这个控制，读者可以根据实际系统的运行情况，考察如何实现这一部分控制，会使

算法的效率更高。

2. 温度 T 的降低,可以采用如下公式:

$$T = \lambda T \tag{6-8}$$

式中 λ 叫做冷却率,一般情况下可以在[0.8,0.9]之间取值。Geman 在 1984 年曾经证明:温度的下降必须与时间的对数成反比,网络最终才能收敛到全局极小点。按照 Geman 的要求,对温度 T,应采用如下公式:

$$T = \frac{T_0}{\log(1 + t)} \tag{6-9}$$

式中的 t 为人工时间,T_0 为温度的初值。

3. T 的初值 T_0 可以按照如下方法设定:

(1) $T_0 = E(\{w^{(h)}\})$;即:取初始系统目标函数(能量)的值。

(2) $T_0 = z E(\{w^{(h)}\})$。即:取初始系统目标函数(能量)值的若干倍。

(3) 按照经验给出。

4. 2.3 中 $w_{ij}^{(p)}$ 的调整量 $\Delta w_{ij}^{(p)}$ 的计算是比较麻烦的,可以根据 Boltzmann 分布或者 Gauss 分布来计算,也可以用其他的方法。下面讨论按 Gauss 分布进行计算的方法。取如下形式的 Gauss 分布函数(简洁起见,用符号 w 代替符号 $w_{ij}^{(p)}$):

$$p(\Delta w) = \exp\left(-\frac{\Delta w^2}{T^2}\right) \tag{6-10}$$

由于算法需要用的是 Δw,而不是 $p(\Delta w)$,所以,必须根据上式按照一定的方法求出 Δw。下面介绍的方法叫做 Monte Carlo 法。

对 $p(x)=\exp\left(-\frac{x^2}{T^2}\right)$ 执行从 0 到 Δw 的积分。但是,由于该函数无法用常规方法进行积分,所以使用数值积分法:

首先,可以根据网络的精度要求,设一个积分步长 δ,然后通过数值积分构造出如下形式的表格。

表 6-1 联接权调整量的数值积分

Δw	δ	2δ	3δ	4δ	…	$N\delta$
$\int_0^{\Delta w} p(x)\mathrm{d}x$	C_1	C_2	C_3	C_4	…	C_N

有了上述表格之后,当需要一个联接权的调整量时,首先按照均匀分布在$[C_1, C_N]$中随机地取一个值 C,然后,从

$$\{C_1, C_2, C_3, \cdots, C_N\}$$

中选取 C_k 满足:

$$|C_k - C| = \min\{|C - C_1|, |C - C_2|, |C - C_3|, \cdots, |C - C_N|\} \tag{6-11}$$

这个 C_k 对应的 $k\delta$ 就是所需要的联接权调整量 Δw。

6.3 Cauchy 训练

在上节给出的模拟退火算法中，使用的是依照 Gauss 分布来确定联接权的调整量的方法。Gauss 分布与 Boltzmann 分布类似，加上其他具有类似特征的一些分布，它们对应的训练被称为 Boltzmann 训练。1987 年，S. Szu 和 R. Hartley 提出用 Cauchy 分布去取代 Boltzmann 分布，Cauchy 分布的一般形式为

$$p(x) = \frac{T}{T^2 + x^2} \qquad 6-12$$

为了叙述方便起见，称按照 Cauchy 分布来确定联接权的调整量的训练为 Cauchy 训练。用 Cauchy 分布替代 Boltzmann 分布，可以从三个方面获得好处。

首先，对于 $[C_1, C_N]$ 中的任意一个 C，它按照 Cauchy 分布所能取到的联接权的调整量要大于按照 Boltzmann 分布所能取到的联接权的调整量。这就是说，同样的条件下，在 Cauchy 训练中，取到较大的联接权调整量的机会要多于 Boltzmann 训练。显然，取到较大的联接权调整量的机会多，训练的速度就快，算法的效率就高。

其次，用 Cauchy 分布取代 Boltzmann 分布后，温度可以下降得更快。这时，温度的下降变得与时间成反比：

$$T = \frac{T_0}{1 + t} \qquad 6-13$$

这与 Boltzmann 训练中的温度的下降必须与时间的对数成反比（式 6－9）相比，进一步大大地提高了训练的速度。

由于 Cauchy 分布是遵从式 6－12 的，而该函数可以用常规的方法进行积分运算：

$$\begin{aligned}\int_0^{\Delta w} p(x)\mathrm{d}x &= \int_0^{\Delta w} \frac{T}{T^2 + x^2}\mathrm{d}x \\ &= T\int_0^{\Delta w} \frac{1}{T^2 + x^2}\mathrm{d}x \\ &= T\left[\frac{1}{T}\arctan\frac{x}{T}\right]_0^{\Delta w} \\ &= \arctan\frac{\Delta w}{T}\end{aligned}$$

即

$$P(\Delta w) = \arctan\frac{\Delta w}{T} \qquad 6-14$$

从而有

$$\tan(P(\Delta w)) = \frac{\Delta w}{T}$$

所以

$$\Delta w = T\tan(P(\Delta w)) \qquad 6-15$$

按照通常的习惯，可以加上学习率 α，此时有：

$$\Delta w = \alpha T\tan(P(\Delta w)) \qquad 6-16$$

由于上式中的 $P(\Delta w)$ 是积分的结果，这使得 Monte Carlo 法在这里变得非常简单了：在(0，1)中按照均匀分布随机地取一个数作为 $P(\Delta w)$，再取当前的温度，就可以直接计算出 Δw。

Cauchy 训练算法可以通过将算法 6－2 中的 Boltzmann 分布换成相应的 Cauchy 分布就可以实现了。

6.4 相关的几个问题

前几节介绍了人工神经网络基本的非确定训练方法，实际上，还有许多相关的问题。本节将讨论其中作者认为比较重要的几个问题。它们包括：Boltzmann 机、人工热问题、BP 算法与 Cauchy 训练的结合问题。目的在于使读者对上述内容能更好地理解和应用。

1. Boltzmann 机

在本章中，介绍了多层网络的 Boltzmann 训练，虽然 Boltzmann 机(BM)也是一种用非确定的方法进行训练的人工神经网络模型，而且 Boltzmann 分布虽然在 Boltzmann 训练和 Boltzmann 机都起着重要的作用，但它们是不同的。

首先，该模型中的神经元与前面叙述的神经元有所不同。在这里，每个神经元可以有一个特殊的阈值，用来限制神经元所获得的激活值。对神经元 AN_j，它所获得的激活值为

$$net_j = \sum_{k=1}^{n} w_{kj}x_k - \theta_j \qquad 6-17$$

如图 6－4 所示。式中的 θ_j 为阈值。这个阈值也可以不存在。在没有阈值的情况下，神经元所获得的激活值为

$$net_j = \sum_{k=1}^{n} w_{kj}x_k \qquad 6-17'$$

神经元的状态按照一定的概率发生变化。$o_j = 1$ 的概率为

$$P_j = \frac{1}{1+\exp\left(-\frac{net_j}{T}\right)} \qquad 6-18$$

其次，从网络的联接结构来看，Boltzmann 机也与前面叙述的基本网络模型有所不同。在 Boltzmann 机中，每对神经元之间都存在着信息的双向传送。一般地，将双向的联接权的大小取为相等的。但是，网络中不允许存在从神经元自身到自身的信息反馈。所以，Boltzmann 机的联接权矩阵是一个对角线元素为 0 的对称矩阵。由于其神经元之间都有联接，所以，有时候又将这种网络称为全互联的。一般用下列式子作为 Boltzmann 机的目标函数(能量函数)：

$$E = -\sum_{k<j} w_{kj}o_k o_j + \sum_{k=1}^{n} o_k\theta_k \qquad 6-19$$

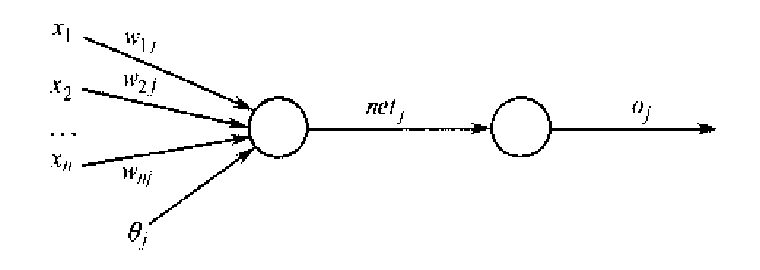


图 6-4 Boltzmann 机中的神经元

将该函数取负，则可以用来表示网络的和谐一致性。所以，有时又将下式称为"一致性函数"：

$$E = -\sum_{k<j} w_{kj} o_k o_j - \sum_{k=1}^{n} o_k \theta_k \qquad 6-20$$

Boltzmann 机神经元之间的全互联，使它成为了一种含有循环的网络。因此，有关 Boltzmann 机的训练算法等问题，将留在介绍循环网络时进行讨论。

2. 人工热问题

虽然 Cauchy 分布给训练取得较大的联接权修改量提供了更多的机会，而且它已经使得温度的下降速度达到了与时间成反比的程度，但是，该训练算法所需要的时间还是很长。因此，继续寻找更快速的算法成为人们探索的方向之一。

无论是 Boltzmann 训练，还是 Cauchy 训练，都是在模拟金属的退火过程。因此，研究者们感到，继续从热力学中寻找适当的技术来进一步提高算法的速度是有可能的。

在仔细研究金属的退火过程中人们发现，该过程表现出"离散的"能量变化的现象。这些离散的能量变化，将退火过程分成一个一个的"阶段"。实际上，在一个阶段到下一个阶段的过渡处，一种特殊的热可能有一个相当大的跃变。这里的特殊热被定义为：温度关于能量的变化率。特殊热的这种变化，可以与系统落入能量的局部极小点相对应。为了防止系统陷入这样的局部极小点，应该在能量从一个阶段跃变到下一个阶段的温度出现时，对网络的"温度"的下降速度加以更加严格的控制。此时的下降速度应该大大地低于网络在其他时刻的"温度"的变化率。我们将系统在其能量跃变的边界处的温度叫做临界温度，系统的临界温度是网络训练中需要特别关注的温度。

与此相对应，作为一个复杂的非线性系统，人工神经网络在训练过程中也会经过相似的阶段。可以依照热力学中特殊热的概念，定义一个人工特殊热(也可以叫做"伪特殊热")。在能量阶段变化的边界点，这个人工特殊热也会有一个非常激烈的变化。其定义为：系统的人工温度关于系统的能量函数(目标函数)的平均变化率。在系统训练的开始阶段(高温期)和较后的阶段(低温期)，能量变化比较平稳，这个时期的人工特殊热几乎为常数。相当于在这两个阶段，系统可能距离局部极小点比较远。因而，在这两个阶段，温度可以变化得快一些，以较好地提高系统的训练速度。

与此相反，当系统处于临界温度处的时候，温度的小量下降，会引起能量函数值的较大变化，相当于系统正处于一个局部极小点附近。因而，算法在这些"关口"应使温度缓慢下降。如图 6

-1 所示，当系统处于局部极小点 A 附近时，如果温度下降过快，它会被陷在 A 点。这样以来，系统就无法达到全局极小点 B。

显然，这类临界温度点是可以通过考察所定义的人工特殊热的变化情况得到的。

3. BP 算法与 Cauchy 训练的结合

Cauchy 训练的速度比 Boltzmann 训练快，但是与 BP 算法相比，它又是比较慢的。人们已经给出的测试结果表明，Cauchy 训练所需花费的时间大约是 BP 算法所需花费时间的 100 倍。其原因在于 BP 算法总是沿着极小化能量函数的方向来调整联接权，而且每次调整都是确定的。然而，在 Cauchy 训练中，权的调整是随机的。同时，由于这种调整的随机性和试探性，还需要判定一个拟进行的调整是否为有效的。所以，在很多时候，所进行的调整是失败的——它可能被放弃。在它未被放弃时，它又可能是背离寻找全局极小点的总目标的。而且，在 Cauchy 训练中，有时可能会获得过大的调整量，可能产生过大的联接权，这又可能使网络瘫痪。

BP 算法的问题是，由于其收敛证明使用的是无穷小的联接权修改量，这种无穷小的修改量就预示着无穷长的训练时间。这是难以用算法实现的，算法实现的只是相应的有穷近似。所以，它的收敛性并没有充分的保证。而且，局部极小点问题、网络的瘫痪问题，也是困扰 BP 算法的两个大问题。

Cauchy 训练的最大长处是，它有可能使网络逃离局部极小点。

一个比较好的办法是将这两种算法结合起来，构成一个新的算法。它既能兼顾到 BP 算法的联接权的调整量是确定的所带来的优点，又能兼顾到 Cauchy 算法中联接权调整(量)的随机性和试探性所带来的可以从局部极小点逃离出来的优点。因此，BP 算法与 Cauchy 算法的结合，可以采用将一个联接权的修改量分成两部分的办法来实现：由 BP 算法提供直接计算部分，由 Cauchy 算法提供随机部分。

对网络中的神经元 AN_i 与 AN_j 之间的联接权 w_{ij}，用下式进行调整：

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

$$\Delta w_{ij} = \alpha((1-\beta)\delta_j o_i + \beta \Delta w'_{ij}) + (1-\alpha)\Delta w_{ij}^{(c)} \qquad 6-21$$

式中 $\Delta w_{ij}^{(c)}$ 为根据 Cauchy 算法所获得的联接权 w_{ij} 的调整量；$\Delta w'_{ij}$ 为 w_{ij} 的上一次的修改量；$\alpha \in (0,1)$ 为学习率，在这里，它同时又起到“直接部分”和“随机部分”的权重分配的作用；$\beta \in (0,1)$，为冲量系数。

顺便应该提到，随着训练的进行，网络的联接权可能会变得很大，使得网络陷入瘫痪。为了解决此问题，当发现神经元的网络输入使它处于饱和的边缘时，应该执行对网络联接权的压缩。例如，如果想将联接权压缩在 $(-a, a)$ 以内，P. D. Wasserman 曾给出如下建议公式：

$$w_{ij} = \frac{2a}{1+\exp\left(-\dfrac{w_{ij}}{a}\right)} - a$$

练　习　题

1. 简单叙述人工神经网络的确定方法和非确定方法，并对它们的特点进行比较。

2. 叙述人工神经网络非确定的训练方法的基本思想。

3. 算法 6－1 的第 10 步中，要求重复执行第 1 步到第 9 步，"直到网络满足要求"。你如何考虑解决与之相关的三个问题：

(1) 以什么样的标准来判断网络是否满足要求？

(2) 在用一个样本对网络的某一个联接权进行修改后(该修改可能是有效的，也可能是无效的)，是随机地抽取另一个联接权进行重复，还是再选择下一个样本进行重复？

(3) 对一个选定的样本，每次是否可以选取若干个联接权进行修改？如果可以，还应做什么工作？

4. 参考图 6－3 所给的分析，弄清楚人们是如何想到将模拟退火用于人工神经网络的训练的。如果可能，请你给出你认为模拟退火被用于人工神经网络训练的更为合理的探索过程。

5. 叙述算法 6－2 的基本思想。

6. 设计出算法 6－2 的实现程序。

7. 给出 Cauchy 训练算法。

8. 指出 Boltzmann 训练与 Boltzmann 机的不同之处。

9. 请在算法 6－2 中加入人工热，实施对训练过程的控制。

10. 请设计一个算法，将 Cauchy 训练与 BP 算法结合起来。

第七章　循环网络

前面四章，讨论的都是非循环网。在非循环网中，信息被从输入端加在网上，通过网络的逐级加工，最后由输出端输出。这个过程中不存在信号的反馈。对这种网络来说，无论输入什么样的信息，它都会给出一个确定的输出，不管这个输出是否是用户所需要的。一般来说，如果输入是被训练时所用的样本集所覆盖的，而且网络的训练是成功的，则它会给出一个用户满意的输出；否则，网络的输出很可能是难以令人满意的。但是，无论怎样，相应的输出总是确定的。

然而在循环网中，网络在接收到一个信号后，它要让这个信号在网络中经过反反复复的循环处理，直到变化停止，或者变化的幅度足够小时，网络在此时刻给出的相应输出才能算是它的输出。如果对给定的输入，网络都能给出一个相应的输出，则称此网是稳定的。如果它在接到信号以后，输出端不断地出现新的信号，也就是说，它的输出总保持有超出给定范围的变化，则认为网络是不稳定的。显然，对一个给定的输入，网络输出的不断变化是由它的反馈信号引起的。

对循环网络来说，理想的状况是：在信号被输入后，信号的变化遵循如下规律：

强烈变化 ⟹ 较弱的变化 ⟹ 不变化

这就是说，网络对输入信号进行的处理是一个逐渐“修复”、“加强”的过程。随着“修复”、“加强”的不断进展，网络输出端的信息逐渐达到用户的要求，此时信号也就不再会有更大的变化。

由于 Hopfield 对循环网络的贡献，因此，在这里将循环网络称为 Hopfield 网。

本章将讨论循环网络的组织、功能与运行算法。包括实现自相联映射的 Hopfield 网络和实现异相联映射的相联存储器。分 7 节介绍如下内容：Hopfield 网络实现的自相联存储；稳定性分析；统计 Hopfield 网与 Boltzmann 机；基本双联存储器的结构与训练；其他的几种相联存储网络。在本章的最后，作为一个例子，介绍如何使用 Hopfield 网去解决一个已经被证明是 NP 的问题——TSP 问题。

7.1　循环网络的组织

在 Hopfield 网络中，每对神经元之间都是互联的，每个神经元都没有到自身的联接。也就是说，Hopfield 网是一个全互联的网络。假设网络的神经元的个数为 h，输入向量的维数为 n，输出向量的维数为 m。对应于输入向量和输出向量，网络中有 n 个神经元被指定为直接接受输入的神经元，简称它们为输入神经元。同时，有 m 个神经元被指定为输出计算结果的神经元，类似地，简称它们为输出神经元。一个网络中的输入神经元和输出神经元是可以重叠的。为了清晰起见，图 7－1 只给出了输入神经元和输出神经元不重叠的情况，输入神经元和输出神经元有重

叠的情况的处理是类似的，相应的讨论完全适应。在此网络中，显然有

$$h \geqslant n, h \geqslant m, n \geqslant 1, m \geqslant 1$$

其中，当

$$n = m = h \qquad 7-1$$

时，网络退化成最简单的形式。称之为最基本的 Hopfield 网络。

在网络的非简单形式下，除了输入神经元和输出神经元外，其他神经元就是该网络的隐藏神经元。当式 7-1 成立时，网络中就不存在隐藏的神经元。

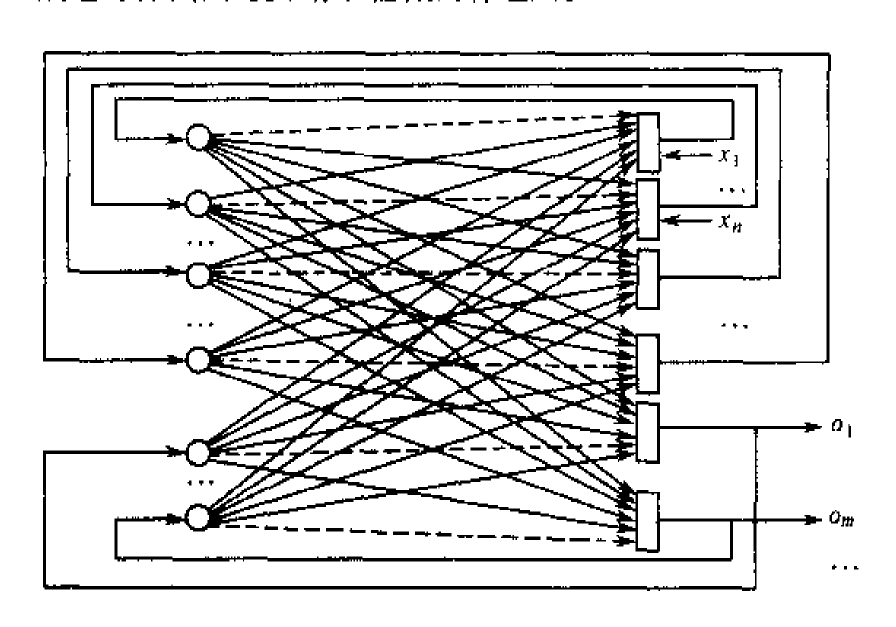


图 7-1　全互联网络

在下面的叙述中，设：网络的神经元 AN_i 与神经元 AN_j 之间的联接权为 w_{ij}，网络的联接权矩阵为：

$$\boldsymbol{W} = \{w_{ij}\}$$

由于网络中的所有神经元都不包含到自身的直接联接，所以，在图 7-1 中，用虚线表示这类联接。相应地，在联接矩阵 $\boldsymbol{W}$ 中，对所有的 i，$w_{ii}=0$ 成立。此时，仍然用 $\boldsymbol{X}$ 和 $\boldsymbol{O}$ 分别表示网络的输入向量和输出向量：

$$\boldsymbol{X} = (x_1, x_2, \cdots, x_n)$$

$$\boldsymbol{O} = (o_1, o_2, \cdots, o_m)$$

对循环网来说，另一个必须确定的问题是，网络中神经元的状态采用的是什么样的变换方式。实际上，与所有其他的人工神经网络模型相同，在上述循环网络中，神经元的状态的变化也有两种不同的形式：第一种，各个神经元的状态的变化是非同步的。在这种方法中，由于神经元的状态是随机离散变化的，所以，研究起来比较复杂；第二种，各个神经元的状态的变化是同步的。在这种方法中，所有的神经元按照同一个系统时钟同步变换状态。因为主要是为了说明这种网络的运行机理，所以，在下面的讨论中，仍然假定网络是按第二种方式运行的。此外，在这里还假定式 7-1 成立，即网络取最简单的形式，即最基本的 Hopfield 网络。并且假定在一个输入

加在网络上后，直到网络给出一个稳定的输出，它一直加在网络上。所以对于网络中的任意一个神经元 $AN_j(1\leqslant j\leqslant n)$，它的网络输入：

$$net_j = \sum_{i=1\&i\neq j}^{n} w_{ij}o_i + x_j \qquad 7-2$$

同时，各神经元的激活函数为如下形式的阈值函数：

$$o_j = \begin{cases} 1 & \text{if } net_j > \theta_j \\ 0 & \text{if } net_j < \theta_j \\ o_j & \text{if } net_j = \theta_j \end{cases} \qquad 7-3$$

式中，θ_j 为神经元 AN_j 的阈值。图 7－2 所示为最基本的 Hopfield 网络。

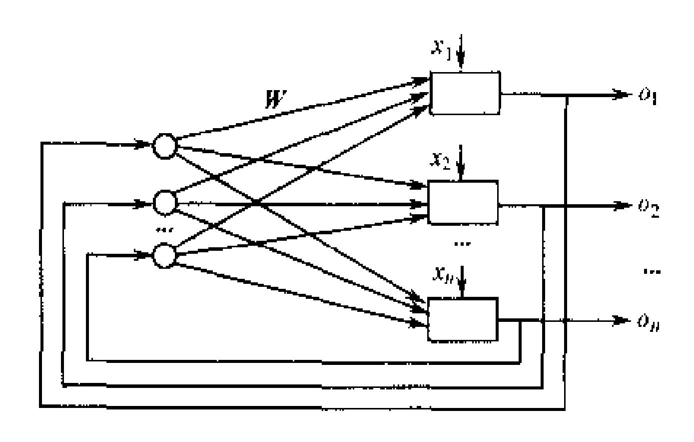


图 7－2　最基本的 Hopfield 网络

希望该网络可以实现联想功能，并且希望在联想过程中实现对信息的"修复"和"加强"。按照这种希望，就可以要求网络的联接矩阵存放的是一组这样的样本：它的输入向量和输出向量是相同的向量，即，$\boldsymbol{X}=\boldsymbol{Y}$。因此可以按如下方法确定它的联接权矩阵：

设网络训练用的样本集为

$$\boldsymbol{S} = \{Y_1, Y_2, \cdots, Y_s\}$$

对 $i\neq j$，取

$$w_{ij} = \sum_{k=1}^{s} y_{ik}y_{jk} \qquad 7-3$$

对 $1\leqslant i\leqslant n$，取

$$w_{ii} = 0 \qquad 7-4$$

式中，y_{ik} 表示 $\boldsymbol{Y}_i$ 的第 k 个元素。下面的矩阵表示形式会使我们有更清楚的理解。取

$$
\boldsymbol{W}_0 = \begin{bmatrix} \sum_{k=1}^{s} y_{k1}^2 & 0 & 0 & \cdots & 0 \\ 0 & \sum_{k=1}^{s} y_{k2}^2 & 0 & \cdots & 0 \\ 0 & 0 & \sum_{k=1}^{s} y_{k3}^2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \sum_{k=1}^{s} y_{kn}^2 \end{bmatrix} \qquad 7-5
$$

则

$$
\boldsymbol{W} = \boldsymbol{Y}_1^T \times \boldsymbol{Y}_1 + \boldsymbol{Y}_2^T \times \boldsymbol{Y}_2 + \cdots + \boldsymbol{Y}_2^T \times \boldsymbol{Y}_s - \boldsymbol{W}_0
$$

$$
= \begin{bmatrix} y_{11}y_{11} & y_{11}y_{12} & \cdots & y_{11}y_{1n} \\ y_{12}y_{11} & y_{12}y_{12} & \cdots & y_{12}y_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ y_{1n}y_{11} & y_{1n}y_{12} & \cdots & y_{n1}y_{1n} \end{bmatrix} + \cdots + \begin{bmatrix} y_{s1}y_{s1} & y_{s1}y_{s2} & \cdots & y_{s1}y_{sn} \\ y_{s2}y_{s1} & y_{s2}y_{s2} & \cdots & y_{s2}y_{sn} \\ \cdots & \cdots & \cdots & \cdots \\ y_{sn}y_{s1} & y_{sn}y_{s2} & \cdots & y_{sn}y_{sn} \end{bmatrix} - \boldsymbol{W}_0
$$

$$
= \begin{bmatrix} y_{11}y_{11} + \cdots + y_{s1}y_{s1} & y_{11}y_{12} + \cdots + y_{s1}y_{s2} & \cdots & y_{11}y_{1n} + \cdots + y_{s1}y_{sn} \\ y_{12}y_{11} + \cdots + y_{s2}y_{s1} & y_{12}y_{12} + \cdots + y_{s2}y_{s2} & \cdots & y_{12}y_{1n} + \cdots + y_{s2}y_{sn} \\ \cdots & \cdots & \cdots & \cdots \\ y_{1n}y_{11} + \cdots + y_{sn}y_{s1} & y_{1n}y_{12} + \cdots + y_{sn}y_{s2} & \cdots & y_{1n}y_{1n} + \cdots + y_{sn}y_{sn} \end{bmatrix} - \boldsymbol{W}_0
$$

由式 7-3 知,对任意的 i 和 $j(i \neq j)$,

$$
w_{ij} = \sum_{k=1}^{s} y_{ik}y_{jk} = \sum_{k=1}^{s} y_{jk}y_{ik} = w_{ji}
$$

所以,$\boldsymbol{W}$ 是一个对角线元素为 0 的对称矩阵。

与前面遇到过的训练方法不同,在这里是根据样本集直接地计算出网络的联接矩阵。显然,这种训练方法效率要高许多。另外,由于 $\boldsymbol{W}$ 是各个样本向量自身的外积的和,所以,有时称该网络实现的是自相联映射。

与所述的二值系统相对应,当将神经元的激活函数改为 s 形函数后,系统就成为一个连续系统。

参照前面讲过的多级网络的组织方式,还可以将循环网络中的神经元分层进行组织,以构成一个多级循环网络。一般地,在一个多级循环网络中,除了输出层的输出向量被反馈到输入层外,其他各层之间的信号传送均执行如下规定:第 $i-1$ 层神经元的输出经过第 i 个连接矩阵被送入第 i 层。一般不考虑越层的信号传送和中间的信号反馈,也不考虑同层的神经元之间进行信号的直接传送,即:在同层内的神经元之间不设置直接的连接。当然,这并不是说这种联接在多层循环网中是不允许的。在特殊的情况下,这种层内神经元之间的直接连接可以用来实现层内神经元之间的竞争,或者层内神经元分组内的相互激活。

有关多级循环网的结构等,将从7.4节开始进行介绍。

7.2 稳定性分析

网络的稳定性是与收敛性不同的问题。如前所述,收敛性是指在网络的训练过程中,算法最终能使网络变得对训练样本集中的所有样本都能给出误差精度要求范围内的结果。而网络的稳定性则是指,在网络的运行中,对一个输入向量,由于网络中信号反馈的作用,经过网络不断地"修复"、"加强",最后能给出对应该输入的适当的输出向量。这是循环网络中特有的问题。如果对一个给定的输入向量,网络无休止地进行"修复"、"加强",而无法给出一个适当的输出,那么就无法使用该网络。因此,稳定性问题是困扰循环网络的一个非常重要的问题。

1983年,Cohen和Grossberg证明了Hopfield网络的稳定性定理:

如果Hopfield网络的联接权矩阵是对角线为0的对称矩阵,则它是稳定的。

下面,按照7.1节所给的假设来证明这个结论。

在前面的章节中讨论网络训练的收敛性问题时,曾经给网络定义了一个能量函数(又被称为目标函数),网络的训练过程被看成是进行网络能量函数的极小化的过程。参考这种做法,我们认为,在Hopfield网络的运行过程中也应该有一个类似的能量函数。如果网络的运行是稳定的,则对任意一个输入向量,在网络的运行过程中,这个目标函数是广义下降的。这样,随着网络的运行,目标函数逐渐减小,最终达到极小点。而当网络运行到它的能量函数达到极小点时,网络的状态就不再发生新的变化,则此时网络的输出就是输入向量相应的输出向量。也就是,将判定网络运行的稳定性问题,转换成了考察网络能量函数是否能够下降到极小点的问题。

可以用著名的Lyapunov函数作为Hopfield网络的能量函数:

$$E = \frac{1}{2}\sum_{i=1}^{h}\sum_{j=1}^{h} w_{ij}o_io_j - \sum_{j=1}^{n} x_jo_j + \sum_{j=1}^{h} \theta_jo_j \qquad 7-6$$

此函数中的各项按照如下的意义表达出关于网络的稳定性度量。

$w_{ij}o_io_j$:该项是网络的一致性测度。当神经元AN_i、AN_j的输出与w_{ij}一致时,它对E的贡献将是一个负数,起到减小能量函数值的作用;而当神经元AN_i、AN_j的输出与w_{ij}不一致时,它对E的贡献将是一个正数,起到增加能量函数值的作用。

事实上,当神经元AN_i、AN_j的输出是同号的时候,它们应该表现出是互相支持的。所以,此时应该有$w_{ij}>0$。这使得$w_{ij}o_io_j>0$,从而给E的贡献是一个负数。如果此时$w_{ij}<0$,表明神经元AN_i、AN_j的输出状态与$w_{ij}<0$表现出的"互相抑制"相矛盾,所以此时它对E的贡献是一个正数。

当神经元AN_i、AN_j的输出是异号的时候,它们应该表现出是互相抑制的。所以,此时应该有$w_{ij}<0$,这使得仍然有$w_{ij}o_io_j>0$,从而给E的贡献还是一个负数。但是,如果此时$w_{ij}>0$,表明神经元AN_i、AN_j的输出状态与网络的要求(它们应该互相抑制)不相符,此时,$w_{ij}o_io_j<0$,从而该项对E的贡献是一个正数。

x_jo_j:表示同一个神经元的输入和它的输出的一致性测度。显然,希望对同一个神经元来说,

它的输出应该与它的输入相一致。实际上,就是希望网络能够消除这些不一致。所以,输入向量中隐藏的这种不一致越少,网络将来需要做的工作就越少,网络就更趋于稳定。

$\theta_j o_j$:这一项是神经元自身的稳定性的测度。θ_j 越高,o_j 越高,表明该神经元在网络中需要接受的网络输入越高,从而要求的网络联接权的值越高,这样,网络就趋向于不稳定。

只要能证明,在网络的运行过程中它的能量函数(式 7-6)总能保证是(广义)下降的,就证明了网络是稳定的。注意到网络的运行表现出来的只是神经元的状态的变化,所以,只要我们能证明对于网络的任意神经元的状态变化,网络的能量函数总是(广义)下降的,就证明了该网络是稳定的。

因此,仍然任意取一个神经元 AN_k,在网络的某一次循环中,它的状态从 o_k 变成 o'_k。按照 Cohen 和 Grossberg 所给的 Hopfield 网络的稳定性定理的要求,设 $\boldsymbol{W}$ 是对角线为 0 的对称矩阵:

$$w_{ij} = w_{ji} \quad (1 \leqslant i \leqslant h, 1 \leqslant j \leqslant h)$$

$$w_{ii} = 0 \qquad (1 \leqslant i \leqslant h)$$

并且设网络在原始状态下的能量函数为

$$E = \frac{1}{2}\sum_{i=1}^{h}\sum_{j=1}^{h} w_{ij}o_i o_j - \sum_{j=1}^{n} x_j o_j + \sum_{j=1}^{h}\theta_j o_j \qquad 7-6$$

则当 AN_k 的状态从 o_k 变成 o'_k 时,有如下两种情况:

(1) AN_k 是输入神经元

AN_k 的状态从 o_k 变成 o'_k 后网络的能量函数 E' 为

$$\begin{aligned} E' = & -\frac{1}{2}\sum_{i=1\&i\neq k}^{h}\sum_{j=1\&j\neq k}^{h} w_{ij}o_i o_j - \sum_{j=1\&j\neq k}^{n} x_j o_j + \sum_{j=1\&j\neq k}^{h}\theta_j o_j \\ & -\frac{1}{2}\sum_{j=1\&j\neq k}^{h} w_{ij}o'_k o_j - \frac{1}{2}\sum_{j=1\&j\neq k}^{h} w_{jk}o_j o'_k - \frac{1}{2}w_{kk}o'_k o'_k \\ & -\frac{1}{2}x_k o'_k + \theta_k o'_k \end{aligned}$$

注意到 $\boldsymbol{W}$ 的对称性,上式中的第 4 项求和与第 5 项求和是相同的,可以将它们合并起来。另外,矩阵 $\boldsymbol{W}$ 的对角线元素为 0,所以,上式中的第 6 项可以删除。因此,有

$$\begin{aligned} E' = & -\frac{1}{2}\sum_{i=1\&i\neq k}^{h}\sum_{j=1\&j\neq k}^{h} w_{ij}o_i o_j - \sum_{j=1\&j\neq k}^{n} x_j o_j + \sum_{j=1\&j\neq k}^{h}\theta_j o_j \\ & -\sum_{j=1\&j\neq k}^{h} w_{kj}o'_k o_j - x_k o'_k + \theta_k o'_k \end{aligned} \qquad 7-7$$

再注意到式 7-6 与 7-7 中不同的项只是那些含有 o'_k 的项,两式相减,可以得到能量的增量 ΔE:

$$\begin{aligned} \Delta E = & E' - E \\ = & -\left[\sum_{j=1\&j\neq k}^{h} w_{kj}(o'_k - o_k)o_j\right] - x_k(o'_k - o_k) + \theta_k(o'_k - o_k) \\ = & -\left[\sum_{j=1\&j\neq k}^{h} w_{kj}o_j + x_k \quad \theta_k\right](o'_k - o_k) \end{aligned}$$

$$= -\left[\sum_{j=1}^{h} w_{kj}o_j + x_k - \theta_k\right](o'_k - o_k)$$

$$= -(net_k - \theta_k)\Delta o_k$$

即

$$\Delta E = -(net_k - \theta_k)\Delta o_k \qquad 7-8$$

其中，Δo_k 表示 AN_k 状态的变化。当

$$\Delta o_k = 0$$

时，有

$$\Delta E = 0$$

由式 7-3 可知，当

$$\Delta o_k > 0$$

时，必有

$$o'_k = 1 \& o_k = 0$$

这表示 o_k 由 0 变到 1，因此必有

$$net_k > \theta_k$$

所以

$$net_k - \theta_k > 0$$

从而

$$-(net_k - \theta_k)\Delta o_k < 0$$

故此时有

$$\Delta E < 0$$

这就是说，网络的目标函数是下降的。同理，读者可以自行讨论当 $\Delta o_k<0$ 时的情况，这里不再详细讨论。

(2) AN_k 不是输入神经元

AN_k 的状态从 o_k 变成 o'_k 后网络的能量函数 E' 为

$$E' = -\frac{1}{2}\sum_{i=1\&i\neq k}^{h}\sum_{j=1\&j\neq k}^{h} w_{ij}o_io_j - \sum_{j=1}^{n} x_jo_j + \sum_{j=1\&j\neq k}^{h}\theta_jo_j$$

$$-\frac{1}{2}\sum_{j=1\&j\neq k}^{h} w_{kj}o_jo'_k - \frac{1}{2}w_{kk}o'_ko_k + \theta_ko'_k$$

与 AN_k 是输入神经元的讨论相同，上式中的第 4 项求和与第 5 项求和可以合并起来，第 6 项可以删除，整理得

$$E' = -\frac{1}{2}\sum_{i=1\&i\neq k}^{h}\sum_{j=1\&j\neq k}^{h} w_{ij}o_io_j - \sum_{j=1}^{n} x_jo_j + \sum_{j=1\&j\neq k}^{h}\theta_jo_j$$

$$-\sum_{j=1\&j\neq k}^{h} w_{kj}o'_ko_j + \theta_ko'_k$$

从而有

$$\begin{aligned}
\Delta E &= E' - E \\
&= -\left[\sum_{j=1\&j\neq k}^{h} w_{kj}(o'_k - o_k)o_j\right] + (o'_k - o_k) \\
&= -\left[\sum_{j=1\&j\neq k}^{h} w_{kj}o_j - \theta_k\right](o'_k - o_k) \\
&= -\left[\sum_{j=1}^{h} w_{kj}o_j - \theta_k\right](o'_k - o_k) \\
&= -(net_k - \theta_k)\Delta o_k
\end{aligned}$$

即

$$\Delta E = -(net_k - \theta_k)\Delta o_k \qquad 7-10$$

这里值得说明的是，在变换中取

$$net_k = \sum_{j=1}^{h} w_{kj}o_j$$

这是因为 AN_k 不是输入神经元，所以，它的网络输入只是 $\sum_{j=1}^{h} w_{kj}o_j$。当然，也可以认为该神经元对应的输入向量的元素的值为 0。在这种假设下，讨论可以得到相同的结论。

用与对式 7－8 类似的方法对式 7－10 进行讨论，可以得到：无论 ANk 的状态是如何变化的，总有：

$$\Delta E < 0$$

综上所述，E 在网络的运行中是广义下降的。

需要说明的是，在上面的推导中用到了"**W** 是对角线为 0 的对称矩阵"这个条件，正是这个条件保证了 Hopfield 网络的稳定性。但是，必须指出，这个条件并不是必要条件。事实上，在许多系统中，其联接权矩阵 **W** 并不一定是对角线为 0 的对称矩阵。当然，有时候，对称的微小偏差也会导致系统的不稳定情况的发生。

7.3 统计 Hopfield 网与 Boltzmann 机

在 6.4 节，我们曾经因为讨论 Boltzmann 训练而提到过 Boltzmann 机。但是在那里，没能讨论它的训练问题。本节将结合循环网络，对 Boltzmann 机的训练问题进行适当的讨论。与此相关，将首先讨论统计 Hopfield 网。

1. 统计 Hopfield 网

在上面介绍的二值 Hopfield 网中，它的每个神经元的状态都是按照式 7－2、7－3 计算确定的。在网络的非确定训练方法中，曾经根据一定的概率来修改神经元之间的联接权，以使网络能脱离能量的局部极小点。而按照上节的分析，Hopfield 网络在其运行过程中，也是在寻找其能量函数的极小点。因此，也可以考虑让网络以统计的方法来确定其各个神经元的状态，并最终达到能量函数的极小点。此外，为使网络的训练能顺利进行，还在非确定的方法中引入了对金属热加

工中的退火的模拟。用类似的方法，在 Hopfield 网络运行中，也让各个神经元的状态的确定概率与一个类似的“人工温度”相关。从而使得 Hopfield 网络的运行实现对金属退火过程的模拟。这就是统计 Hopfield 网。

在统计 Hopfield 网络中，每个神经元的状态与它所获得的网络输入、阈值、温度相关：

$$p_k = \frac{1}{1 + \exp\left(-\frac{net_k - \theta_k}{T}\right)} \tag{7-11}$$

式中：net_k——AN_k 所获得的网络输入；

θ_k——AN_k 的阈值；

T——系统的人工温度；

p_k——表示 AN_k 的状态取 1 的概率。

在这里，当系统面对一个输入向量开始寻找它的相应输出时，它所具有的人工温度所取的初始值应该是很大的。而且，每个输入神经元都应该取相应的输入向量的分量所规定的状态。然后，网络按照如下算法去寻找系统能量函数的极小点。

算法 7-1 统计 Hopfield 网运行算法

1 取一个很大的值作为人工温度 T 的初值；

2 对网络中每一个神经元 AN_k，

 2.1 按照式 7-11 计算相应的概率 p_k；

 2.2 按照均匀分布，在[0,1]中取一个随机数 r；

 2.3 如果 $p_k > r$ 则使 AN_k 的状态为 1，否则使 AN_k 的状态为 0；

3 逐渐降低温度 T，如果温度足够低，则算法结束；否则，重复 2。

上述算法中，人工温度的初值、温度的下降率都可以参考“非确定方法”一章中的有关介绍，这里不再重复。

2. Boltzmann 机的训练

Boltzmann 机是一种多级循环人工神经网络，可以看成是 Hopfield 网的一种扩展。具体可以参考第 6 章。

我们用最速下降法对 Boltzmann 机进行训练。设任意神经元 AN_i 的阈值表示为 θ_i，w_{ij} 表示神经元 AN_i 到 AN_j 的联接权。o_i、o_j 分别是神经元 AN_i 与 AN_j 的实际输出状态。与统计 Hopfield 网络中的神经元相同，$o_i = 1$ 的概率为

$$p_i = \frac{1}{1 + \exp\left(-\frac{net_i - \theta_i}{T}\right)}$$

当人工温度 T 趋近于 0 时，每个神经元的状态不再具有随机性，而只具有给定的特性。此时，Boltzmann 机退化成一般的 Hopfield 网。

Boltzmann 机的能量函数用式 7－12 表示。由于该式同时还表达出了网络联接权的和谐一致性，所以又被称为该网络的一致性函数。

$$E = -\sum_{i<j} w_{ij} o_i o_j - \sum_{j=1}^{h} \theta_j o_j \qquad 7-12$$

现在考察在 Bolztmann 机的运行中每次只有一个神经元的状态发生改变的情况。设神经元 AN_i 在这次运行中状态发生了变化。由式 7－12 可知：

$$\begin{aligned}\Delta E_i &= E(o_i = 0) - E(o_i = 1) \\ &= \sum_j w_{ij} x_j - \theta_j \end{aligned} \qquad 7-13$$

根据系统应该选择具有能量较低的状态的原理，此时，如果 $\Delta E_i > 0$，则应该选 AN_i 输出为 1，否则，应该选 AN_i 输出为 0。

从概率的角度来看，ΔE_i 的值越大，神经元 AN_i 应该处于状态 1 的概率就应该越大。反之，ΔE_i 的值越小，神经元 AN_i 应该处于状态 1 的概率就应该越小。从而，$o_i = 1$ 的概率为

$$p_i = \frac{1}{1 + \exp\left(-\frac{\Delta E_i}{T}\right)} \qquad 7-14$$

设 Boltzmann 机有两个状态 a、b，它们分别对应于 $o_i = 1$ 和 $o_i = 0$，而系统中其他的神经元在这两个状态下的状态是不变的。分别用 P_a 和 P_b 表示系统出现这两个状态的概率。显然，它们是与 p_i 相关的，取

$$P_a = \gamma p_i = \gamma \frac{1}{1 + \exp\left(-\frac{\Delta E_i}{T}\right)} \qquad 7-15$$

$$P_a = \gamma(1 - p_i) = \gamma \frac{\exp\left(-\frac{\Delta E_i}{T}\right)}{1 + \exp\left(-\frac{\Delta E_i}{T}\right)} \qquad 7-16$$

从而

$$\begin{aligned}\frac{P_a}{P_b} &= \exp\left(\frac{\Delta E_i}{T}\right) \\ &= \exp\left(\frac{E(o_i = 0) - E(o_i = 1)}{T}\right) \\ &= \exp\left(-\frac{E_a - E_b}{T}\right)\end{aligned}$$

即

$$\frac{P_a}{P_b} = \exp\left(-\frac{E_a - E_b}{T}\right) \qquad 7-17$$

这就是 Boltzmann 分布。由此可见，当网络进行足够多次的迭代后，它处于某一状态的概率既取决于该网络在此状态下的能量，又取决于此时系统的温度。由于高温时网络的各个状态出

现的概率基本相同，这就给它逃离局部极小点提供了机会。

当系统的温度较低时，对式 7-17 进行分析，不难得出，如果 $E_a > E_b$，则 $P_a < P_b$。这说明，此时网络处于较低能量状态的概率较大。

那么，如何按照统计的方法对 Boltzmann 机进行训练呢？1986 年，Hinton 和 Sejnowski 给出了一种方法，通过两个神经元的“约束概率”和“自由概率”来计算它们之间的联接权的调整量：设网络在没有输入的自由状态下，神经元 AN_i 和 AN_j 同时处于激发状态的概率为“自由概率”P_{ij}^{-}。网络在加上输入后神经元 AN_i 和 AN_j 同时处于激发状态的概率为“约束概率”P_{ij}^{+}。他们认为，神经元 AN_i 和 AN_j 之间的联接权的修改量与相应的“自由概率”和“约束概率”有关。其关系由式 7-18 表示：

$$\Delta w_{ij} = \alpha(P_{ij}^{+} - P_{ij}^{-}) \qquad 7-18$$

他们还将 Boltzmann 机的训练算法分成三大步骤，如算法 7-2。

算法 7-2　Boltzmann 机训练算法

1　计算约束概率

　1.1　对样本集中的每个样本，执行如下操作：

　　1.1.1　将样本加在网络上(输入向量及其对应的输出向量)；

　　1.1.2　让网络寻找平衡；

　　1.1.3　记录下所有神经元的状态。

　1.2　计算对所有的样本，AN_i 和 AN_j 的状态同时为 1 的概率

2　计算自由概率

　2.1　从一个随机状态开始，不加输入、输出，让网络自由运行，并且在运行过程中多次纪录网络的状态；

　2.2　对所有的 AN_i 和 AN_j，计算它们的状态同时为 1 的概率 P_{ij}^{+}。

3　对权矩阵进行调整

　用 $\Delta w_{ij} = \alpha(P_{ij}^{+} - P_{ij}^{-})$ 对 w_{ij} 进行修改。

算法中，P_{ij}^{+} 表示网络在样本集中的所有样本的约束下，神经元 AN_i 和 AN_j 的状态同时为 1 的概率；P_{ij}^{-} 表示网络在没有样本约束的自由状态下，神经元 AN_i 和 AN_j 的状态同时为 1 的概率。

算法的 1.1.2 步让网络寻找平衡是指，当将一个样本加到网络上以后，网络开始寻找能量函数的极小点，当它到达这个极小点时，它就处于平衡状态。从表现形式上看，此时网络的输出不再变化。在此过程中，可以加进人工热，使它在温度逐渐降低的过程中寻找能量的极小点。

算法的 1.1.3 步要求记录下网络中所有神经元的状态。这是因为在下一步计算 P_{ij}^{+} 时，要对所有的 AN_i 和 AN_j 计算它们同时输出 1 的次数以及其他情况的次数。例如，如果样本集中共有样本 s 个，网络对这 s 个样本的运行结果是 AN_i 和 AN_j 有 c 次同时输出为 1，则此时：

$$P'_{ij} = \frac{c}{s} \qquad 7-19$$

值得注意的是，这里需要的是神经元 AN_i 和 AN_j 的状态同时为 1 的概率，实现时需要对每一个样本存储一次网络状态（每个神经元在网络处于能量函数的极小点时的输出值）。这样才有利于找出神经元 AN_i 和 AN_j 的状态同时为 1 的次数，以计算出相应的概率。P^-_{ij} 的求法类似。

7.4 双联存储器的结构

在第 1 章曾经提到，人类的记忆通常都具有联想的功能。我们都有这样的经验：当我们获得某一项成果而使心情非常好时，会自然地联想起以前曾经为此做出的努力，曾经在某时某地有过怎样的美好经历……当我们遇到某个挫折时，马上就会认真地检讨自己，看自己曾经在哪些方面出现了漏洞，有过什么样的失误。当我们闲下来时，会从今天的环境、状态、条件等联想到未来。研究者认为，所有这种现象，是因为人脑具有这样的机制：它可以从某一件事出发，给予一个所谓的智力链，从一件事想到另一件事。这使得人们有能力去“唤回失去的记忆”。

在前面三节中，已经看到了人工神经网络对联想功能的模拟。在那里，网络主要是实现对模式的“修补”、“加强”。实际上，在网络中，存放着若干个模式，对于一个待回忆的内容，网络通过对用户提供的“线索”进行联想，最终找到和这个“线索”匹配最好的模式。所以，它实现的联想是属于自相联的。

在另一些时候，需要网络能够在接收到一个模式后，将该模式与不同的模式联系起来。但是上一章所介绍的网络模型却无法实现这个功能，这是由它的单层结构造成的，因为在单层结构上，可以看成网络的输入和输出是出现在同一组神经元上，所以，它实现的是自身到自身的映射。

为了解决不同模式的联想，必须使用不同的神经元组去对应网络的输入向量和输出向量。这样，对应加在一组神经元上的一个输入模式，网络就可以在另一组神经元上产生一个与之相关联的不同输出。这种联想被称为异相联。实现这种联想的网络叫做双联存储器（Bidirectional Associative Memory——简记为 BAM）。

此外，与 Hopfield 网类似，双联存储器还具有一定的推广能力。它对含有一定缺陷的输入向量，通过对信号的不断变换、修补，最后给出一个正确的输出。

本节将首先介绍双联存储器的结构。

图 7-3 给出的是最基本的双联存储器。实际上，它是一个双层循环网络。为了清楚地表现出输入向量经过联接矩阵 $\boldsymbol{W}$（第 2 层的联接权矩阵）被传送到第 2 层神经元，在第 1 层神经元后加了一个“联接”，该“联接”就象第 2 层神经元的输出通过“第 0 层”的联接后经权矩阵 $\boldsymbol{W}^T$（第 1 层的联接权矩阵）送到第 1 层一样。

在该网络中，信号的传送过程是这样的：输入向量 $\boldsymbol{X}$ 被通过联接权矩阵 $\boldsymbol{W}$ 传送到下一层，此时在网络的第 2 层上产生输出向量 $\boldsymbol{Y}$，输出向量 $\boldsymbol{Y}$ 又被通过（反馈）联接矩阵 $\boldsymbol{W}^T$ 反馈到第 1 层，该层又产生出新的向量 $\boldsymbol{X}$，$\boldsymbol{X}$ 又被通过联接权矩阵传送到第 2 层，此时又产生新的输出向量 $\boldsymbol{Y}$……这个过程被循环下去，直到网络的输出 $\boldsymbol{Y}$ 不再变化为止。

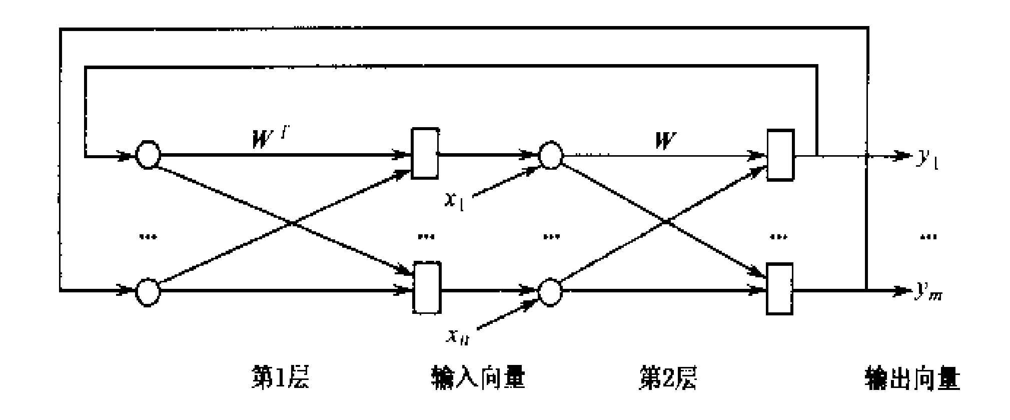


图 7-3　基本的双联存储器结构

图 7-3 中的神经元所执行的功能与多级网中其他神经元执行的功能是相同的。网络运行可用下面的式子加以描述：

$$\boldsymbol{Y} = \boldsymbol{F}(\boldsymbol{XW}) \qquad 7-20$$

$$\boldsymbol{X} = \boldsymbol{F}(\boldsymbol{YW}^T) \qquad 7-21$$

其中，输入、输出向量 $\boldsymbol{X}$、$\boldsymbol{Y}$ 是如下形式的向量：

$$\boldsymbol{X} = (x_1, x_2, \cdots, x_n)$$

$$\boldsymbol{Y} = (y_1, y_2, \cdots, y_m)$$

$\boldsymbol{F}$ 为神经元的激活函数，一般可采用 s 形函数：

$$y_i = \frac{1}{1 + \exp(-\lambda net_i)}$$

随着式中 λ 值的不断增加，该函数趋近于阈值为 1 的阈值函数。所以，在简单情况下，可以取

$$y_i = \begin{cases} 1 & \text{if } net_i > 0 \\ 0 & \text{if } net_i < 0 \\ y_i & \text{if } net_i = 0 \end{cases} \qquad 7-22$$

式中，当神经元的网络输入为 0 时，它的状态保持不变。

依照上述运行方式，对一个给定的输入向量，网络的输出向量应该由开始的较大变化逐渐过渡到最后的不变化。当然，这是网络的理想运行情况。Kosko 在 1987 年已经证明，基本的双联存储器是无条件稳定的。这是因为它的第 1 层和第 2 层的联接权矩阵是互为转置矩阵。

显然，当输入向量的维数与输出向量的维数相同时，$\boldsymbol{W}$ 就变成了一个方阵，此时如果联接矩阵 $\boldsymbol{W}$ 是对称的：

$$\boldsymbol{W} = \boldsymbol{W}^T$$

则由式 7-20 和 7-21 所进行的变换是完全相同的，而两个完全相同的变换可以用网络的两次循环来完成。此时，基本的双联存储器就退化成了一个 Hopfield 网。

7.5 异相联存储

对 Hopfield 网,我们说它实现了自相联的映射,它的权矩阵按照式 7-3 和 7-4 计算设定。双联存储器实现的是异相联映射,因此,可以考虑用类似的方法,通过计算来设定联接权矩阵 $\boldsymbol{W}$。

设网络的训练样本集为

$$S = \{(\boldsymbol{X}_1, \boldsymbol{Y}_1), (\boldsymbol{X}_2, \boldsymbol{Y}_2), \cdots, (\boldsymbol{X}_s, \boldsymbol{Y}_s)\}$$

取

$$W = \sum_{i=1}^{s} X_i^T Y_i \qquad 7-23$$

作为联接权矩阵。

一个值得注意的问题是,对于任何一个网络来说,它的容量是有限的。在不超过网络的存储容量的情况下,按照此式构造出来的双联存储器,可以实现所要求的变换。当样本集中的输入向量被送到网络中的时候,网络可以立即给出相应的结论。

在下列两种情况下,网络需要对输入向量进行反复的循环处理,才能给出输出结果:

(1) 当输入向量中含有"噪音"时,网络需要在反复的循环过程中逐渐地将所含的噪音去掉。当噪音被去掉后,网络的输出就是该输入向量对应的适当输出,此时网络达到能量函数的极小点,这使得网络的状态不再继续发生变化。

(2) 当样本集所含的信息超出网络的容量时,即使给出的输入向量是样本集中的某一个向量,网络也会进入循环,而且在网络运行结束后,它给出的结果也可能并不是正确的结果。

关于网络的容量,有如下一些结果:

(1) 1987 年,Kosko 给出过一个较为悲观的估计。他认为,在一般情况下,相联存储器的容量不会超过网络的最小层的神经元的个数。

(2) 1988 年,Haines 和 Hecht - Nielson 证明,如果双联存储器中的每个神经元各选一个阈值,构成一个"非均匀的(Nonhomogenous)"网络,则网络的存储容量最多可以达到 2^{min},其中,min 为网络的最小层的神经元的个数。这种非均匀的网络是基本双联存储器的扩充。在这里,式 7-22 被换成

$$y_i = \begin{cases} 1 & \text{if } net_i > \theta_i \\ 0 & \text{if } net_i < \theta_i \\ y_i & \text{if } net_i = \theta_i \end{cases} \qquad 7-22'$$

按照 Haines 和 Hecht - Nielson 的证明,通过分别给网络中的每一个神经元选择适当的阈值,可以使得该网络的稳定状态数取 1 与 2^{min} 之间的任何数。然而,由于用户所要求系统的稳定状态并不是随机选取的,它们是要根据问题的实际决定的。所以,实际上,系统的容量很难达到这个数据。R. J. McEliece、E. C. Posner、E. R. Rodemich 和 S. S. Venkatesh 关于 Hopfield 网络的存储容量的结论被 P. D. Wasserman 用于双联存储器的容量估计。结论指出,如果用户随机地

选择 L 个状态，并且要求每个向量中有 $4+\log_2 min$ 个分量为 1，其他为 -1，则可以构造出一个非均匀的双联存储器，它可以使 98% 的向量成为稳定状态。而且要求

$$L < 0.68 \frac{0.68 min^2}{(\log_2 min + 4)^2}$$

不幸的是，对输入向量中 1 的个数的限制，严重地影响了它的应用。尤其重要的是，它的运行可能达不到稳态。也就是说，网络可能产生不了适当的结果。

虽然双联存储器的容量有较大的局限性，但是，从网络的结构及其训练来看，它又具有非常简单、容易实现等特点。

7.6 其他的双联存储器

1. 连续的双联存储器

在基本的双联存储器中，假定网络中神经元的状态是同步变换的，而且这些神经元使用的都是阈值函数。这使得网络的表现能力受到一定的限制。而且在生物神经系统中，这两点都是不存在的。仔细对其进行分析发现，这两点限制主要是为讨论方便而设的。取消这两点限制后，网络将有更强的表达能力。Kosko 在 1987 年已经证明，取消这两个限制后，双联存储器仍然是稳定的。

2. 自适应双联存储器

前面，对 Hopfield 网和双联存储器都是用计算的方法进行训练的。除此之外，也可以让双联存储器在运行中调整联接权。最简单的方法是使用 Hebb 学习律进行训练，即对任意 w_{ij}，取：

$$\Delta w_{ij} = \alpha o_i o_j \qquad 7-24$$

3. 具有竞争的双联存储器

在双联存储器中，可以通过在层内神经元之间添加附加的联接来实现竞争。这些权构成另一个主对角线元素为正值，其他元素为负值的权矩阵。Cohen - Grossberg 定理指出，如果权矩阵是对称的，则网络是稳定的。

实际上，即使权矩阵不是对称的，网络通常也是稳定的。然而，目前还不知道哪一类权矩阵会引起不稳定。

7.7 Hopfield 网用于解决 TSP 问题

循环网络有很多应用，其中，1985 年，J. J. Hopfield 和 D. W. Tank 巧妙地将此技术用于 TSP 问题的求解。这种方法虽然难以给出 TSP 问题的最优解，但是，试验表明，当城市的个数不超过 30 时，在多数时候，它可以给出最优解的近似解。而当城市的个数超过 30 时，最终的结果就不太理想了。尽管如此，它还是表明了循环网络的能力。与此同时，作者认为，J. J. Hopfield 和 D. W. Tank 用循环网络来描述 TSP 问题的方法，是独具匠心的，非常值得学习。因此，用这一结果的描述作为本章的结束。

所谓 TSP 问题,简单地讲,就是:一个售货员要顺序地访问 n 个城市,并最终回到起点。城市之间均存在给定距离的通路,希望选择的路径是最短的。在有 n 个城市的情况下,将存在$\frac{n!}{2n}$条可能的路径。例如,当 $n=60$ 时,可能的路径就有 69.34155×10^{78} 条。显然,不可能一条一条地去计算它们的长度,最后再从中选择出最短的路径来,因为这个计算量太大了。人们已经证明,这个问题是一个 NP 问题。

设问题中含有 n 个城市。因为要用神经元的状态来表示出这 n 个城市被访问的顺序,所以,对应一个城市需要有 n 个神经元。这相当于说,该网络将含有 n^2 个神经元。"可以"将这些神经元排成一个 $n\times n$ 的方阵。让该方阵的每一行对应一个城市。当该行的第几个神经元的输出是 1 时,该城市就被第几个访问。所以,正常的情况下,在运行过程中,网络应该保持这样的状态:方阵的每行神经元中有且仅有一个输出为 1;同时,该方阵的每列神经元中也有且仅有一个输出为 1。

表 7-1 给出了四个城市的 TSP 问题的表示。分别称这四个城市为 X、Y、Z、W。如果从城市 X 出发,则它表现出来的访问顺序为:第 1 先访问城市 X,第 2 访问城市 W,第 3 访问城市 Y,第 4 访问城市 Z,最终返回到城市 X。

表 7-1 四个城市的 TSP 问题的表示

城市名	被访问的顺序			
	1	2	3	4
X	0	1	0	0
Y	0	0	0	1
Z	1	0	0	0
W	0	0	1	0

对后面用到的符号,约定如下:

d_{xy}——城市 X 与城市 Y 之间的距离;

y_{xi}——城市 X 的第 i 个神经元的状态:

$$y_{xi}=\begin{cases}1 & \text{城市 } X \text{ 在第 } i \text{ 个被访问}\\0 & \text{城市 } X \text{ 不在第 } i \text{ 个被访问}\end{cases} \qquad 7-25$$

$w_{xi,yj}$——城市 X 的第 i 个神经元到城市 Y 的第 j 个神经元的连接权。

按下列公式设置连接矩阵:

$$w_{xi,yj}=-A\delta_{xy}(1-\delta_{ij})-B\delta_{ij}(1-\delta_{xy})-C-\zeta d_{xy}(\delta_{ji+1}+\delta_{ji-1}) \qquad 7-26$$

其中

$$\delta_{ij}=\begin{cases}1 & \text{如果 } i=j\\0 & \text{如果 } i\neq j\end{cases} \qquad 7-27$$

式 7-26 的第 1 项 $-A\delta_{xy}(1-\delta_{ij})$ 中,当 $X\neq Y$ 时,因 δ_{xy} 为 0 而使该项为 0,此时使得 $-A$ 不起作用;而当 $X=Y$ 且 $i\neq j$ 时,δ_{ij} 就为 0,此时该项取值 $-A$,用来实现同一行中的神经元之间

的互相抑制。所以，可以认为，$-A$ 为列内神经元之间的抑制因子。

式 7-26 的第 2 项 $-B\delta_{ij}(1-\delta_{xy})$中，当 $i\neq j$ 时，因 δ_{ij}为 0 而使该项为 0，此时使得 $-B$ 不起作用；而当 $i=j$ 且 $X\neq Y$ 时，δ_{ij}不为 0，而此时 δ_{xy}为 0，这使得该项取值 $-B$，用来实现同一列中的神经元之间的互相抑制。所以，可以认为，$-B$ 为行内神经元之间的抑制因子。

式 7-26 的第 3 项 $-C$ 则是对任意 X、Y、i、j 都发挥作用的。所以，可以将它看成是全局抑制因子。

式 7-26 的第 4 项 $\zeta d_{xy}(\delta_{ji+1}+\delta_{ji-1})$为距离项。其中，$\zeta$ 为城市间的距离在连接权中的反映因子。显然，该项只有在 δ_{ji+1}与 δ_{ji-1}中有一个不为 0 时才发挥作用，此时要么 $i+1=j$，要么 $i-1=j$。注意到 $w_{xi,yj}$是表示 X 是被第 i 个被访问的，Y 是被第 j 个被访问的情况，所以，当 $i+1=j$ 时，表示访问 X 后立即访问 Y；而当 $i-1=j$ 时，表示访问 Y 后立即访问 X。无论是哪一种情况，售货员旅行的距离都是 d_{xy}，此时它被加到相应的权上。

网络的能量函数被定义为

$$
\begin{aligned}
E = &\frac{A}{2}\sum_{x}\sum_{i}\sum_{j\neq i}y_{xi}y_{xj} \\
&+\frac{B}{2}\sum_{i}\sum_{x}\sum_{z\neq x}y_{xi}y_{zi} \\
&+\frac{C}{2}\Big(\sum_{x}\sum_{i}y_{xi}-n\Big)^2 \\
&+\frac{D}{2}\sum_{x}\sum_{z\neq x}\sum_{i}d_{xz}y_{xi}(y_{zi+1}+y_{zi-1})
\end{aligned}
\qquad 7-28
$$

其中的第 1 项仅当所有的城市最多只被访问一次时取得极小值 0，每当一个城市被重复访问一次，能量就增加$\frac{A}{2}$，所以，A 为城市被多次访问的惩罚因子。

第 2 项仅当每次最多只访问一个城市时取得极小值 0。如果出现多个城市被同时访问的现象，表明是发生了不可能的情况，因此必须抑制。一次访问的城市每多一个，能量就增加$\frac{B}{2}$。所以，B 为多个城市被同时访问的惩罚因子。

第 3 项当且仅当所有的 n 个城市一共被访问 n 次时才取得最小值 0。C 为一个城市被访问多次，或者系统中存在城市未被访问时的惩罚因子。本项与前两项一起用于保证每个城市被访问且仅被访问一次。

第 4 项表示按照当前的访问路线的安排所需要走的路径的总长度。对每一个城市 X，如果它是被第 i 个访问的话，则当另一个城市 Z 无论是被第 $i+1$ 个访问（紧接着 X 后被访问）还是被第 $i-1$ 个访问（访问城市 Z 后，立即访问城市 X），都要在路径的总长度中加上这两个城市之间的距离 d_{xy}，从而使得当每个城市被访问且仅被访问一次时，选择的路径越优，能量函数的值就越小。

可以按照式 7-26 将 n 个城市之间的距离 d_{xz}写入网络的连接权中，给网络中的每个神经元

一个适当的初值，然后让网络自己去寻找它的能量函数的极小点。当它达到这个极小点时，网络的状态就应该不再发生变化，此时网络的状态给出的就是访问 n 个城市的最佳路径的一个近似解。

练 习 题

1. 什么是循环网络的稳定性？它与网络训练中的收敛问题有什么不同？

2. 试分析循环网络中神经元状态的同步变化和异步变化在网络训练和运行中的不同。变化方式的不同，是否会影响到网络的性能？为什么？另外，对“异步变化”，在训练控制上需要做哪些处理？

3. 试分析用 Lyapunov 函数作为 Hopfield 网络的能量函数来对该网络的稳定性进行评价的合理性。

4. 为什么说“联接权矩阵是对角线为 0 的对称矩阵”是 Hopfield 网络稳定的充分条件？

5. 给出 Hopfield 网络运行算法的思想。

6. 给出 Boltzmann 机训练算法的思想。

7. BAM 是如何实现异相联映射的？如果要实现自相联映射，应该使用哪一个网络模型？

8. BAM 是实现异相联映射的。但是，用 BAM 是否可以实现自相联映射呢？为什么？

9. 循环网是如何实现对 TSP 问题的表示和求解的？

10. 总结非循环网和循环网在训练、运行、功能等方面的特点，指出它们的区别。

第八章 自适应共振理论

在前面介绍过的人工神经网络模型，都是用一个给定的样本集进行训练。在完成训练后，投入正常的运行。如果在使用的过程中环境发生了变化，则需要重新构造一个能够表现当前环境的样本集，并用该样本集重新对网络进行训练。在这种情况下，不能只用变化后新加的样本对网络进行"补充"训练。这会破坏掉网络原来已学会的内容，而只记下新的内容。这就是说，网络的长期存储内容只是它最后获得训练时系统所面对的样本集所蕴含的内容。网络的这种不能在保留已学内容的前提下增加新内容的特性叫做网络的不可塑性。图8－1给出了这一类网络面对环境的变化，所需要采取的策略。

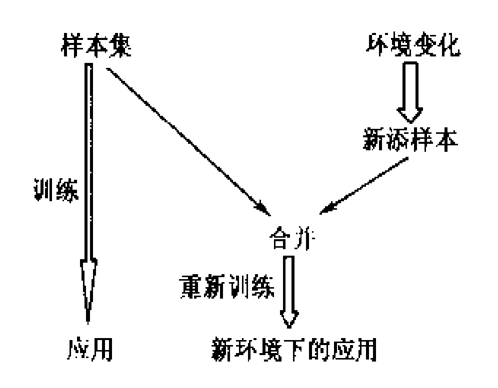


图8－1 环境的变化要求重新训练网络

初步想来，上述情况难以充分地发挥人工神经网络的信息分布存放特性所带来的优点。因为信息分布存放的最大特性之一是信息的部分破坏不会影响所存放的内容。那么在遇到新的变化时，应该可以根据新的样本"部分地"改变原有存储信息，来对网络实施"补充"训练。但试验表明，这是不行的。究竟为什么不能只将代表新添内容的样本拿来对网络实施"补充"训练呢？通过进一步仔细的分析知道，当用代表新添内容的样本来训练网络时，网络的每一个联接权都要根据这些样本进行适当的调整。这样一来，进行的就不是"部分修改"，而是全部的修改了。这种做法的实际效果是：将当前的网络联接权矩阵当作初值，按照"新的样本集"进行新的训练。所以，训练后的网络绝大多数只能表达出"新的样本集"所含的内容。

从上述分析可以得出这样的启示：在面对一些表示新添内容的样本时，如果真正能够只修改一部分内容，这样才有希望在保证不破坏原存储内容的基础上将新的内容增加进去。因此，选择哪一部分进行修改是网络要解决的问题。显然，被修改的内容应该是和新的内容相关的。这就告诉我们，在将样本中所含的内容存入网络的时候，不能再像一般网络那样，实施完全的分布存放。如果要存放也应该是将有关的内容放在一起。也就是说，存放是分类的。

样本所含内容的分类及其存放应该是由网络自动完成的，而且这个过程应该是逐步进行的——要在网络的应用过程中逐渐丰富。也就是说，网络可以"边工作、边学习"，在实践中不断地丰富自己的知识。这种性质称为网络的可塑性。

综上分析，要想使网络具有可塑性，必须实现以下4项功能：

(1) 对样本的分类功能；

(2) 分类的识别功能；

(3) 比较功能；

(4) 类的建立功能。

如果网络具备了这四项功能，它就能够在使用过程中不断地感知大量的信息，从中抽取有价值的内容，对它们进行分类加工，并在不破坏原有存储信息的前提下，存储新的内容——不断丰富自己的知识，增加自己的功能。这正好与人脑对信息的处理相对应。人类在实践中不断丰富自己的知识，人类的大脑可以完成对来自周围环境的大量连续感知信息的分类工作，从这些大量的信息中抽取有价值的东西，对它们进行适当的加工，并在完成加工后，将它们存储起来。在这个过程中，不会因为学习了新的东西而忘掉已学会的东西。也正是因为这一点，才使人类具有非常强的不断地适应新环境的能力，才能不断地生产、不断地丰富自己的物质生活和精神生活。因此，要想使人工神经网络能够发挥更大的作用，也必须使它具有可塑性，有能力适应不断变化的世界的需要。

实际上，Carpenter 和 Grossberg 在 1986 年曾经给出了一个非常特殊的例子：他们只给网络有限的4个样本，这4个样本被周期性地提交给网络，网络需要通过不断地修改各个联接权去适应这个变化的环境。网络联接权的不断变化，表明网络是难以收敛的。后来，Grossberg 及其助手们构造出了具有可塑性的网络模型，叫做自适应共振理论(Adaptive Resonance Theory，简记为ART)。按照其输入数据及其处理，这种网络模型可以分成两大类：一类只接受二值的输入向量；另一类则可以接受二值的和连续值的输入向量。习惯上，人们将前一类叫做 ART1，而将后一类叫做 ART2。当然，可以通过用二进制进行编码，去表达一个非二值表达的数据。但是，从一般的角度来讲，第二种模型的应用范围要更为广泛一些。

本章将介绍 ART1 网络模型。为叙述简洁起见，直接将其记为 ART。主要内容包括：ART 模型的总体结构，各模块功能；比较层的联接矩阵与识别层的联接矩阵的初始化，识别过程与比较过程，查找的实现；ART 的训练。

8.1 ART的结构

为了使网络在保持原有内容的前提下(称之为稳定性)，能够将新的内容添加进去(即可塑性)，按照上述分析，ART 首先必须是一个分类器，它能够将输入向量进行适当的分类，给分类处理与存储打下基础。对一个给定的输入向量，ART 将在网络中已经存放的所有分类中进行查找，如果能够发现其中的某一个"类表示"表达了该输入向量的基本特征，则可对此分类的表示模式进行适当的微调，使之能更好地表达该输入向量。由于被调整的"对象"是已被网络确认的

用来表示输入向量所在类的“类表示”，所以它不影响已有的其他类的“类表示”，这使得网络的稳定性得到保证。对该输入向量，如果网络发现在已有的“类表示”中不存在相对应的，则在自己的容量范围内创造一个新的“类表示”，使它与该输入向量实现匹配。从而使网络具有可塑性。

新输入向量与现存模式 { 相似：修改相匹配的模式 ; 不相似：建立一个新模式 } 不匹配的现存模式不被修改

图 8－2　网络的稳定性与可塑性保证

为实现图 8－2 提出的要求，可以构造出图 8－3 所示的 ART 总体结构图。ART 模型主要包含 5 个功能模块：识别层、比较层、识别层输出信号控制（$G1$）、比较层输出信号控制（$G2$）、系统复位控制。它的基本工作过程为：当系统没有接受输入向量的时候，比较层输出信号控制 $G1$ 使得比较层的输出信号 $\boldsymbol{C}$ 为 0；识别层的输出控制信号 $G2$ 使得识别层的输出信号 $\boldsymbol{P}$ 为 $\boldsymbol{0}$。当输入向量 $\boldsymbol{X}$ 一旦被加到系统上，$G1$ 使 $\boldsymbol{X}$ 被原封不动地按照 $\boldsymbol{C}$ 的形式送入识别层。在识别层找到 $\boldsymbol{C}(\boldsymbol{X})$ 应该属于的类，该类的代表向量被以向量 $\boldsymbol{P}$ 送回到比较层，$\boldsymbol{P}$ 与 $\boldsymbol{X}$ 比较，形成新的输出向 $\boldsymbol{C}$，$\boldsymbol{C}$ 和 $\boldsymbol{X}$ 又同时被送到系统复位控制模块进行比较。如果系统认为 $\boldsymbol{C}$ 可以代表 $\boldsymbol{X}$，则网络进入训练期——按照 $\boldsymbol{X}$ 修改被选中的 $\boldsymbol{B}_k$ 和 $\boldsymbol{T}_k$。如果系统认为 $\boldsymbol{C}$ 不能代表 $\boldsymbol{X}$，则发出信号，使识别层复位（重新输出 0），向量 $\boldsymbol{X}$ 重新被原样送入比较层，寻找新的类进行匹配……如此下去，直到找到一个能满足要求的类或者发现系统中现有的类均不能满足要求。当后一种情况发生时，则在系统中按照 $\boldsymbol{X}$ 建立一个新类。

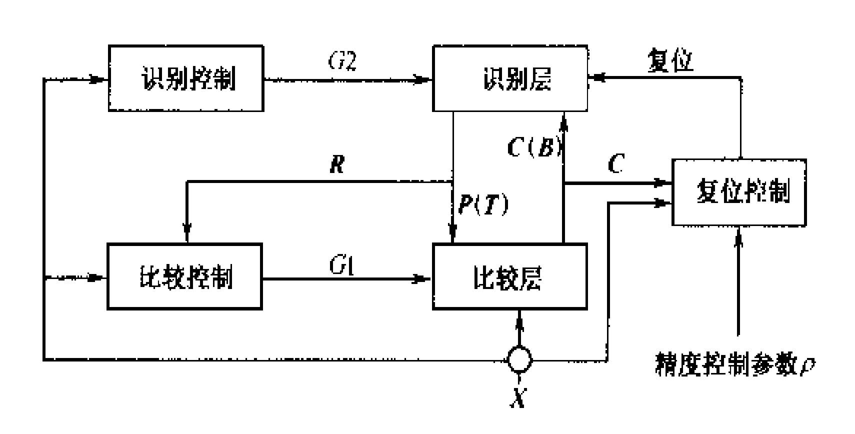


图 8－3　ART 总体结构图

从这个简要的工作过程来看，在系统的 5 个模块中，识别层输出信号控制 $G1$（简称为识别控制）、比较层输出信号控制 $G2$（简称为比较控制）、系统复位控制等 3 个模块是用来实现信号的传输控制的。识别层、比较层承担系统的主要功能，系统中存放的分类信息也由这两层来分析和记忆。因此，为了后面的分析方便起见，图 8－4 给出了 ART 模型的以识别层和比较层为主的拓扑结构，对其他 3 个模块，只标出其控制信号。

在图 8－3、8－4 中，$\boldsymbol{X}$ 为输入向量，$\boldsymbol{R}$ 为识别层的输出向量，$\boldsymbol{C}$ 为比较层的输出向量，$\boldsymbol{P}$ 是

比较层的网络输入向量。后面会看到，$\boldsymbol{P}$ 相当于向量 $\boldsymbol{X}$ 所处的类的表示形式。$\boldsymbol{T}$(Top－down)为从识别层到比较层的联接权矩阵，$\boldsymbol{B}$(Bottom－up)为从比较层到识别层的联接权矩阵，$\boldsymbol{T}_i$ 和 $\boldsymbol{B}_i$ 分别是识别层的第 i 个神经元对应的识别层的联接权向量和比较层的联接权向量：

$$\boldsymbol{X} = (x_1, x_2, \cdots, x_n)$$

$$\boldsymbol{R} = (r_1, r_2, \cdots, r_m)$$

$$\boldsymbol{C} = (c_1, c_2, \cdots, c_n)$$

$$\boldsymbol{P} = (p_1, p_2, \cdots, p_n)$$

$$\boldsymbol{T}_i = (t_{i1}, t_{i2}, \cdots, t_{in})$$

$$\boldsymbol{B}_i = (b_{1i}, b_{2i}, \cdots, b_{ni})$$

其中，t_{ij} 表示识别层的第 i 个神经元到比较层的第 j 个神经元的联接权，b_{ij} 表示比较层的第 i 个神经元到识别层的第 j 个神经元的联接权。p_i 为比较层的第 i 个神经元的网络输入：

$$p_i = \sum_{j=1}^{m} r_j t_{ji} \qquad 8-1$$

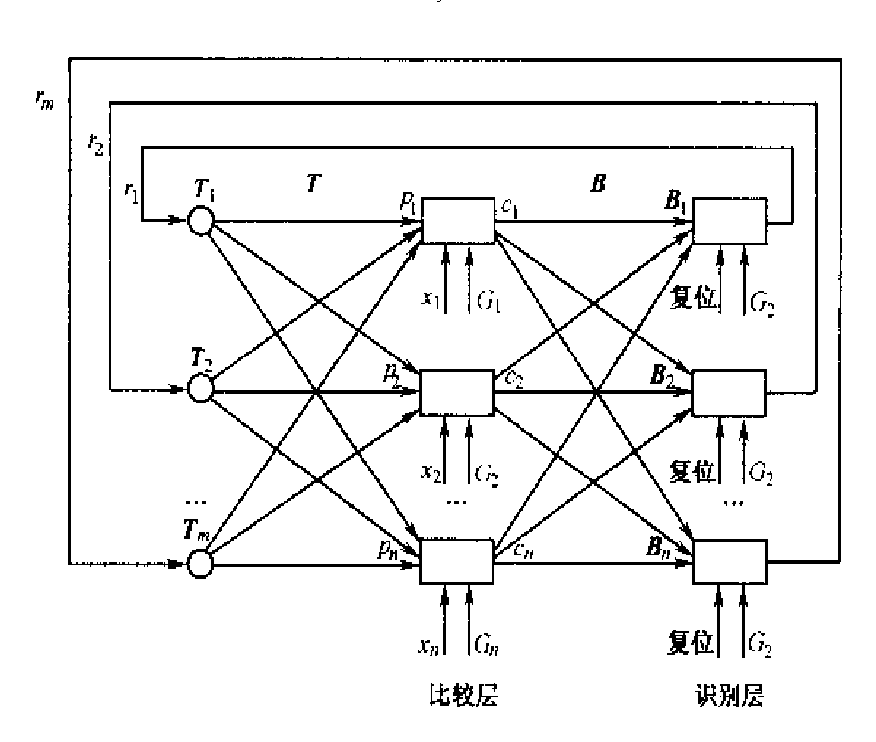


图 8-4　以比较层和识别层为主的 ART 拓扑结构

下面分别讨论这 5 个功能模块。

1. 比较层输出信号控制

由图 8－3 知，比较层输出信号控制模块接收来自外界的输入向量 $\boldsymbol{X}$ 和来自识别层的输出向量 $\boldsymbol{R}$。它根据这两个信号，决定输出 $G1$：仅当 $\boldsymbol{X}$ 为非 0 向量，并且 $\boldsymbol{R}$ 为 0 向量时，$G1$ 才输出 1，其他情况下它均输出 0：

$$G_1 = \neg (r_1 \vee r_2 \vee \cdots \vee r_m) \wedge (x_1 \vee x_2 \vee \cdots \vee x_n) \qquad 8-2$$

也就是说，仅当一个输入向量刚加在网络上（$\boldsymbol{X}\neq\boldsymbol{0}$），识别层对应的输出向量还未出现时（此时 $\boldsymbol{R}=\boldsymbol{0}$，进而使 $\boldsymbol{P}=\boldsymbol{0}$），$G1$ 才输出 1。这是用来配合比较层的工作，使得在网络处理某个输入向量的初始阶段，该输入向量能够被原原本本地送入识别层进行其类别的识别。

2. 识别层输出信号控制

识别层输出信号控制可以看成是识别层的封锁与使能控制。当外界没有信号输入时，$\boldsymbol{X}=\boldsymbol{0}$，这时它封锁识别层，使它输出0 向量（$\boldsymbol{R}=\boldsymbol{0}$、$\boldsymbol{P}=\boldsymbol{0}$）；而在正常运行中（$\boldsymbol{X}\neq\boldsymbol{0}$），它使能识别层，使之可以根据当前的向量 $\boldsymbol{C}$，产生相应于 $\boldsymbol{C}$ 的类表示 $\boldsymbol{P}(\boldsymbol{R})$。

$$G2 = x_1 \vee x_2 \vee \cdots \vee x_n \qquad 8-3$$

3. 比较层

比较层同时接受三个信号：输入信号 $\boldsymbol{X}$，识别层的输出信号 $\boldsymbol{P}$，比较层输出控制信号 $G1$。该层的神经元执行二－三规则。它的第 i 个神经元同时接受 x_i、p_i、$G1$，相应的输出为：

$$c_i = \begin{cases} 1 & x_i + p_i + G1 \geqslant 2 \\ 0 & x_i + p_i + G1 < 2 \end{cases} \qquad 8-4$$

在比较层开始接受一个非0 输入向量 $\boldsymbol{X}$ 之前，称此时网络处于待命期。由于此时相当于网络的输入向量 $\boldsymbol{X}=\boldsymbol{0}$，所以，识别层受到控制信号 $G2$ 的作用，所有的神经元被抑制，此时它的输出向量 $\boldsymbol{R}=\boldsymbol{0}$。当一个非0 的输入向量 $\boldsymbol{X}$ 加在网络上后，网络进入第 1 个工作周期，此时，比较层输出信号控制单元首先工作。由式 8－2，$G1=1$，而此时仍然有 $\boldsymbol{R}=\boldsymbol{0}$（从而使 $\boldsymbol{P}=\boldsymbol{0}$），根据式 8－4，$\boldsymbol{X}$ 被原封不动地传送到识别层：

$$\boldsymbol{C} = \boldsymbol{X} \qquad 8-5$$

当这个 $\boldsymbol{C}$ 被送入识别层的同时，网络的非 0 输入 $\boldsymbol{X}$ 使 $G2=1$（式 8－4），从而使能识别层。识别层中的某一个神经元输出 1，其他输出为 0。不妨设该神经元为第 k 个神经元。由式

$$\begin{aligned} p_i &= \sum_{j=1}^{m} r_j t_{ji} \\ &= r_k t_{ki} \\ &= t_{ki} \end{aligned} \qquad 8-6$$

从而有：

$$\boldsymbol{P} = \boldsymbol{T}_k \qquad 8-7$$

因此，识别层的第 k 个神经元对应的从自身到比较层的各个神经元的联接权所构成的向量是输入向量的类表示。由于 $\boldsymbol{X}\neq\boldsymbol{0}$，所以，在正常情况下，$\boldsymbol{R}\neq\boldsymbol{0}$，$\boldsymbol{P}\neq\boldsymbol{0}$ 成立。由式 8－2，此时 $G1$ 被置为 0。到此时，网络工作的第 1 个周期结束。此时网络进入第 2 个工作周期。按照上述分析，在网络工作的第 2 个周期开始时，有 $G1=0$，$G2\neq0$，$\boldsymbol{R}\neq\boldsymbol{0}$。根据式 8－4，在这个周期，比较层的输出向量 $\boldsymbol{C}$ 实际上是向量 $\boldsymbol{X}$ 和 $\boldsymbol{P}$ 的“与”：

$$c_i = x_i \wedge p_i \quad 1 \leqslant i \leqslant n \qquad 8-8$$

由此可以看出，如果网络认为某一组输入向量是同一类的，则代表它们所在类的向量应该为它们的“与”。

4. 识别层

如图 8-3、8-4 所示，识别层接受三个信号：$G2$、复位、$\boldsymbol{C}$。由式 8-3，当没有输入向量加在网上时，$\boldsymbol{X}=\boldsymbol{0}$，使得 $G2$ 输出 0。这样，识别层的所有神经元被封锁。而当有输入向量时（$\boldsymbol{X}\neq 0$），识别层的神经元被使能。此后，对该输入向量 $\boldsymbol{X}$，按网络的运行，确定自己的状态。

向量 $\boldsymbol{C}$ 是 $\boldsymbol{X}$ 经过比较层变换后输出的向量，它是系统给 $\boldsymbol{X}$ 的"暂定"代表。在比较层的讨论中，我们知道，在 $\boldsymbol{X}$ 被网络处理的第 1 个工作周期中，$\boldsymbol{C}=\boldsymbol{X}$，此时识别层的第 k 个神经元 RN_k 所获得的网络输入为：

$$\sum_{i=1}^{n} b_{ik}c_i \qquad 8-9$$

由于该层的任务是识别出 $\boldsymbol{X}(\boldsymbol{C})$ 应该是属于哪一类的向量，所以，该层实行的是竞争机制。在最简单的情况下，对应于一个向量 $\boldsymbol{C}$，识别层中有且仅有一个神经元处于激发态（输出 1），其他则均处于抑制态。显然，处于激发态的 RN 对应的权向量 $\boldsymbol{B}_k$ 将与 $\boldsymbol{C}$ 匹配的最好。由式 8-9 知，$\boldsymbol{B}_k$ 与 $\boldsymbol{C}$ 有最大的点积：

$$\sum_{i=1}^{n} b_{ik}c_i = \max\left\{ \sum_{i=1}^{n} b_{ij}c_i \,\middle|\, 1 \leqslant j \leqslant m \right\} \qquad 8-10$$

因此，与 $RN_1, RN_2, \cdots, RN_m$ 相对应，向量 $\boldsymbol{B}_1, \boldsymbol{B}_2, \cdots, \boldsymbol{B}_m$ 代表着不同的分类。虽然 $\boldsymbol{X}$、$\boldsymbol{C}$ 是二值向量，但 $\boldsymbol{B}_1, \boldsymbol{B}_2, \cdots, \boldsymbol{B}_m$ 则是实数向量，表明从比较层到识别层的联接权是实数。与之对应的二值表示是比较层相应神经元对应的从识别层到比较层的联接权向量 $\boldsymbol{T}_1, \boldsymbol{T}_2, \cdots, \boldsymbol{T}_m$。$\boldsymbol{B}_1, \boldsymbol{B}_2, \cdots, \boldsymbol{B}_m$ 与 $\boldsymbol{T}_1, \boldsymbol{T}_2, \cdots, \boldsymbol{T}_m$ 依照 $RN_1, RN_2, \cdots, RN_m$ 相对应，从形式上看，下标相同的向量形成一一对应。

识别层的竞争机制是通过各个 RN 之间的侧联接实现的。这些侧联接，除到自身的联接为刺激（正）联接外，其他均为抑制（负）联接。这样，获得最大网络输入的 RN 能够抑制同层其他神经元的激发。

5. 系统复位控制

复位模块也同时接受三个信号：输入（原始）向量 $\boldsymbol{X}$，向量 $\boldsymbol{C}$，精度控制参数 ρ。该模块根据一定的规则计算 $\boldsymbol{C}$ 与 $\boldsymbol{X}$ 的相似度，如果该相似度满足精度控制参数 ρ 的要求，就表示 $\boldsymbol{C}$ 确实可以表示 $\boldsymbol{X}$。如果该相似度不能满足精度控制参数 ρ 的要求，就表示 $\boldsymbol{C}$ 实际上不能表示 $\boldsymbol{X}$。此时，复位控制模块发出复位信号。对应该输入向量 $\boldsymbol{X}$，在不允许 RN_k 激发的条件下，重新寻找它应该对应的类的表达向量。

对一个输入向量 $\boldsymbol{X}$，网络经过运行，找到一个向量 $\boldsymbol{C}$，由上面的讨论知道，它被认为是 $\boldsymbol{X}$ 所处的类的代表。但是，这个代表可能不能满足精度要求。因此，需要继续寻找能满足精度要求的 $\boldsymbol{C}$。在这种情况下，复位控制信号使得本次被"误选"的神经元被屏蔽。在理想的情况下，这个过程仅在输入向量 $\boldsymbol{X}$ 第一次被加在网络上时出现。在 $\boldsymbol{X}$ 被网络认定为某一类后，当它再一次被加在网络上时，网络应该能迅速地找到它所对应的类，而不需要进行反复的查找。一般用下列公式计算 $\boldsymbol{X}$ 与 $\boldsymbol{C}$ 的相似度：

$$s = \frac{\sum_{i=1}^{n} c_i}{\sum_{i=1}^{n} x_i} \qquad 8-11$$

再注意到式 8-11,可以得到 $0 \leqslant s \leqslant 1$。按照精度控制的要求,如果 $s \geqslant \rho$,这表示网络最终认可当前处于激发态的 RN_k 所对应的 $\boldsymbol{B}_k$、$\boldsymbol{T}_k$ 为 $\boldsymbol{X}$ 的类表示。如果 $s < \rho$,表示本次选中的 RN_k 所对应的 $\boldsymbol{B}_k$、$\boldsymbol{T}_k$ 不能很好地代表 $\boldsymbol{X}$,因此,需要重新寻找。

8.2 ART 的初始化

由于 ART 是在运行过程中通过自适应地进行联接权的调整来实现学习的,因此,ART 的初始化非常重要。与前面所叙述的各类网络模型不同,ART 的初始化与它的基本工作原理紧密相关,直接关系到 ART 的可塑性、分类表示等方面的问题,因此需要给予专门的讨论。ART 的初始化主要包括对比较层到识别层的权矩阵 $\boldsymbol{T}$、识别层到比较层的权矩阵 $\boldsymbol{B}$、精度控制参数 ρ 的初始化。下面,分别对其进行讨论。

8.2.1 T 的初始化

$\boldsymbol{T}$ 是从识别层到比较层的联接矩阵,按照上节对识别层和比较层运行的介绍,在网络以最简单的方式工作的情况下,对每一个输入向量,识别层有且仅有一个神经元(如:RN_k)输出 1,其他均输出 0。而在比较层,每个神经元是按照二 - 三规则工作的,由此来实现向量 $\boldsymbol{X}$ 与 $\boldsymbol{P}$ 的比较。由式 8-6、8-7 知道,此时的 $\boldsymbol{P}$ 就是 RN_k 对应的向量 $\boldsymbol{T}_k$。这表明,连接矩阵 $\boldsymbol{T}$ 应该是一个 0、1 矩阵。

对输入向量 $\boldsymbol{X}$,通过 RN_k 被激发,$\boldsymbol{T}_k$ 被以 $\boldsymbol{P}$ 的形式送入比较层。式 8-8 的向量形式为:

$$\boldsymbol{C} = \boldsymbol{X} \wedge \boldsymbol{P} = \boldsymbol{X} \wedge \boldsymbol{T}_k \qquad 8-12$$

而系统的复位控制模块是按式 8-11 来计算 $\boldsymbol{C}$ 与 $\boldsymbol{X}$ 的相似度的,能够保证 $c_i = 1$ 的必要条件为 $x_i = 1$,即对于任意 $1 \leqslant i \leqslant n$,

$$\text{if } c_i = 1 \text{ then } x_i = 1$$

另外,根据 8.3 节中关于 T 的训练的讨论(式 8-17),应该用 1 初始化矩阵 $\boldsymbol{T}$ 的所有元素,即对于任意 $1 \leqslant i \leqslant n$,$1 \leqslant j \leqslant m$,有

$$t_{ij} = 1 \qquad 8-14$$

所以,最开始时,矩阵 $\boldsymbol{T}$ 的所有元素全为 1。

8.2.2 B 的初始化

在上述讨论过程中,对矩阵 $\boldsymbol{B}$ 是按列向量讨论的:$\boldsymbol{B}$ 的列向量 $\boldsymbol{B}_1, \boldsymbol{B}_2, \cdots, \boldsymbol{B}_m$ 依次对应于 $RN_1, RN_2, \cdots, RN_m$,从而每一个向量 $\boldsymbol{B}_k$ 又是 RN_k 对应的类的代表向量。按照网络的运行方式,对于一个输入向量 $\boldsymbol{X}$,如果 RN_k 在竞争中获胜,则 $\boldsymbol{B}_k$ 与 $\boldsymbol{X}$ 的点积取最大值。所以,为了使

网络的运行保持有一定的准确性,需要用较小的值去初始化 $\boldsymbol{B}$ 的每个元素。否则,对同一个输入向量 $\boldsymbol{X}$ 的两次不同的加载,可能会因引起不同的 RN 的激发而导致错误。按照 Grossberg 与 Carpenter 的研究结果,对于任意 $1\leqslant i\leqslant n$,$1\leqslant j\leqslant m$,有

$$b_{ij} < L/(L-1+n) \qquad 8-15$$

其中,n 为输入向量的维数;L 为一个大于 1 的常数,其值应该与输入向量的位数相关。由上式可知,对于任意 $1\leqslant i\leqslant n$,$1\leqslant j\leqslant m$,$b_{ij}\in(0,1)$,是一个非负纯小数。从而,与 0、1 矩阵 $\boldsymbol{T}$ 对应的矩阵 $\boldsymbol{B}$ 是一个实数矩阵。实际上,关于识别层的一个神经元 RN_k、$\boldsymbol{T}_k$、$\boldsymbol{B}_k$ 分别是它对应的类的两种不同形式的表示。另外,由式 8-15,在网络开始运行时,$\boldsymbol{B}$ 的每个元素都是相同的,他们将随着网络的运行逐渐被改变。

8.2.3 ρ 的初始化

顾名思义,精度控制参数 ρ 用来控制网络的识别精度。注意到式 8-11、8-12,我们知道:

$$0\leqslant s\leqslant 1$$

显然,当 $s=0$ 时,本次所得的匹配是最差的;当 $s=1$ 时,是最为理想的匹配。此时,$\boldsymbol{C}=\boldsymbol{X}$ 成立。

由上述分析知,$\rho\in[0,1]$。ρ 的值越大,网络所实现的划分就越细;ρ 的值越小,网络所实现的划分就越粗。这就是说,ρ 的值是划分"粗细"的标准。所以,它的初始化要根据网络的用户要求来确定。也可以在网络运行的初期将 ρ 的值取得小一点,以实现较粗的划分,以后逐渐加大,以实现更精确的划分。

8.3 ART 的实现

本节讨论 ART 的实现。由于 ART 本身具有自适应能力,所以,ART 的训练是在运行过程中根据执行的结果确定的。

在完成初始化后,ART 就可以投入运行,实现"边学习,边工作"。为了叙述清楚起见,将 ART 面对一个输入向量 $\boldsymbol{X}$ 的处理分为四个阶段:识别、比较、查找、训练。

1. 识别

当输入向量 $\boldsymbol{X}$ 未被加在网上时,网络的输入相当于是 0。根据式 8-3,$G2=0$,这使得识别层的所有神经元被抑制。此时,

$$\boldsymbol{R}=(r_1,r_2,\cdots,r_m)=(0,0,\cdots,0)$$

当一个非$\boldsymbol{0}$ 向量被加在网络上时,由式 8-2、8-3,得

$$G1=G2=1$$

成立。而由于此时 $\boldsymbol{R}=\boldsymbol{0}$,所以,根据式 8-1,有

$$\boldsymbol{P}=(p_1,p_2,\cdots,p_m)=(0,0,\cdots,0)$$

再根据式 8-4,获得式 8-5 的结果。此时,在识别层,对于每个 k,$1\leqslant k\leqslant m$,RN_k 完成如下操作:

(1) 计算$\sum_{i=1}^{n} b_{ik}c_i$；

(2) 接收来自其他 RN 的抑制信号，并向其他的 RN 发出抑制信号；

(3) 确定自己的输出状态；

(4) 完成输出。

RN 之间的抑制信号是通过它们之间的抑制连接实现的。如果某一个 RN_k 输出 1，则表明，在本轮识别中，$\boldsymbol{X}$ 暂时被认为是属于该 RN_k 所对应的类。

2. 比较

当识别层在本轮识别中将 $\boldsymbol{X}$ 归于 RN_k 对应的类后，RN_k 的输出值 1 被分别以权重 t_{kj} 传送到比较层，由式 8-6、8-7，此时送入比较层的向量 $\boldsymbol{P}$ 就是向量 $\boldsymbol{T}_k$。$\boldsymbol{T}$ 的初始化及后面讨论的 $\boldsymbol{T}$ 的训练保证了 $\boldsymbol{T}$ 的每个元素取值为 0 或者 1。所以，在前面曾经说，根据 RN_k 进行对应，$\boldsymbol{B}_k$ 与 $\boldsymbol{T}_k$ 互为变换形式。由于此时 $\boldsymbol{R}\neq\boldsymbol{0}$，使得 $G1=0$。由二－三规则，c_i 的值根据式 8-8 确定。如果对于所有的 $j,1\leqslant j\leqslant n,p_j=x_j$，则表示 $\boldsymbol{X}$ 获得良好的匹配。如果存在 j，使得 $p_j\neq x_j$，则表明 $\boldsymbol{X}$ 与相应的"类"的代表向量并不完全一致。

当系统复位控制模块接收到 $\boldsymbol{X}$ 和 $\boldsymbol{C}$ 后，就计算它们的相似度 s：

• 如果 $s\geqslant\rho$，则表明识别层在本轮所给出的类满足系统的精度要求。所以，查找成功，系统进入相应的训练周期。

• 如果 $s<\rho$，则表明识别层在本轮所给出的类不能满足系统的精度要求。此时，复位模块向识别层发出复位信号，使所有的 RN 输出 0。系统回到开始处理 $\boldsymbol{X}$ 的初态，重新进行搜索。由于复位信号在使所有的 RN 输出 0 的同时，屏蔽了本次被激发的 RN，所以，在下一轮的匹配中，该 RN 被排除在外，以便系统能够找到其他更恰当的 RN。

3. 查找

如果 $s\geqslant\rho$，认为网络查找成功，此时分类完成，无需再查找。

如果 $s<\rho$，则表明本轮实现的匹配不能满足要求，此时需要寻找新的匹配向量。具体过程如下：

(1) 复位模块向识别层发出复位信号；

(2) 所有的 RN 均被抑制：$\boldsymbol{R}=(r_1,r_2,\cdots,r_m)=(0,0,\cdots,0)$，上轮被激发的 RN 被屏蔽；

(3) $G1$ 的值恢复为 1；

(4) $\boldsymbol{X}$ 的值再次被从比较层送到识别层：$\boldsymbol{C}=\boldsymbol{X}$；

(5) 不同的 RN 被激发，使得不同的 $\boldsymbol{P}(\boldsymbol{T}_k)$ 被反馈到比较层；

(6) 比较层进行相应的比较，并判定本次匹配是否满足要求；

(7) 如果本次匹配不成功，则重复(1)～(6) 直到如下情况之一发生：

1) 本轮匹配成功。表明已找到一个与 $\boldsymbol{X}$ 匹配较好的模式，此时，网络进入训练期，对这个匹配的模式进行适当的修改，使它能更好地表示 $\boldsymbol{X}$。

2) 网络中现存的模式均不匹配。这表明，$\boldsymbol{X}$ 不属于现存的任何一个类。因此，网络需要重新构造一个新模式来表达这个类。此时，网络用一个还未与任何类关联的 RN 来对应 $\boldsymbol{X}$ 所在的

类:根据 $\boldsymbol{X}$ 修改与此 RN 对应的 $\boldsymbol{T}_k$、$\boldsymbol{B}_k$。由于在进行网络的初始化时,已将 $\boldsymbol{T}$ 的每一个元素初值为 1,而在训练中网络不修改未被选中的连接权向量(见后面的讨论),所以,此时被网络选中的 RN(不妨仍用 RN_k 表示)所对应的从识别层到比较层的连接权向量 $T_k=(1,1,\cdots,1)$。因而,$\boldsymbol{P}=(1,1,\cdots,1)$被送入比较层。由二-三规则,此时 $\boldsymbol{C}=\boldsymbol{X}\wedge\boldsymbol{P}=\boldsymbol{X}$,被送入系统复位控制模块。由式 8-11,$s=1$,而 $\rho\leqslant 1$,所以 $s\geqslant\rho$。匹配获得成功,网络进入训练期。

从上述查找过程看出,当将一个输入向量加到网络上后,网络并不一定能立即找出它所在的类。这是一个值得讨论的问题。因为按照一般的情况,对应输入向量 $\boldsymbol{X}$,由式 8-10,首先被选中的 RN 应该是获得了最大的激励值,为什么输入向量 $\boldsymbol{X}$ 不一定属于 RN 所对应的类呢?分析发现,由于受 $\boldsymbol{B}$ 的值的取法的影响,有时候,获得最大激励值的 RN 对应的类不一定是 $\boldsymbol{X}$ 所属的类。这使得查找成为网络工作的一个必不可少的过程。例如:设 $n=5$,三个输入向量为:

$$\boldsymbol{X}_1=(1,0,0,0,0)$$

$$\boldsymbol{X}_2=(1,0,0,1,1)$$

$$\boldsymbol{X}_3=(1,0,0,1,0)$$

按照式 8-15,假定用 $2/(2-1+5)$ 初始化 $\boldsymbol{B}$,当 $\boldsymbol{X}_1$、$\boldsymbol{X}_2$ 被输入时,RN_1、RN_2 分别被激发,按照式 8-16 对网络进行的训练,$\boldsymbol{T}_1$、$\boldsymbol{T}_2$、$\boldsymbol{B}_1$、$\boldsymbol{B}_2$ 分别取如下值:

$$\boldsymbol{T}_1=(1,0,0,0,0),\quad \boldsymbol{B}_1=(1,0,0,0,0)$$

$$\boldsymbol{T}_2=(1,0,0,1,1),\quad \boldsymbol{B}_2=(0.5,0,0,0.5,0.5)$$

此时,当 $\boldsymbol{X}_3$ 被输入系统时,RN_1、RN_2 获得的激励值都是 1,这两个神经元都有可能被选中而处于激发状态。如果 RN_2 被选中,则此时由比较层输出的向量 $\boldsymbol{C}=\boldsymbol{X}$,从而使得 $s=1$,表明该选择满足网络的精度要求。但是,如果首先选中的是 RN_1,此时比较层的输出向量 $\boldsymbol{C}=(1,0,0,0,0)$,从而使得 $s=0.5$,当 $\rho>0.5$ 时(如取 $\rho=0.8$),选择 RN_1 就不能满足精度要求,此时网络就需要进入查找工作阶段:

(1) RN_1 获胜;

(2) $\boldsymbol{C}$ 取值(1,0,0,0,0);

(3) $s=\sum_{i=1}^{5}c_i/\sum_{i=1}^{5}x_i=0.5$;

(4) $s<\rho$;

(5) RN_1 被屏蔽;

(6) 网络进入第二个查找周期,RN_2 获胜;

(7) $\boldsymbol{C}$ 取值(1,0,0,1,0);

(8) $s=\sum_{i=1}^{5}c_i/\sum_{i=1}^{5}x_i=1$;

(9) 满足精度要求,停止查找,进入训练期。

上述讨论是基于在式 8-15 和式 8-16 取 L 为 2 的前提下进行的,当 L 取其他的值时,如 1.5,4 等,将会有不同的结果。将这些留给读者自己去讨论。

另外,在具体的实现中,读者需要注意,对一个给定的输入向量 $\boldsymbol{X}$,当一个被初步选中的 RN

被系统认为是不能满足精度要求后，在网络继续为该输入向量寻找新的匹配类的过程中，应该一直被屏蔽。要注意的另一个问题是，由于网络中包含着五个主要的功能模块，它们之间互相影响，加上信号的反馈，使得网络中的信号较为复杂。所以，建议读者按照这里提到的"查找周期"的概念去处理信号的传递问题。

四、训练

当网络进入训练期时，已知输入向量与 RN_k 对应的存储模式（向量 $\boldsymbol{T}_k$、$\boldsymbol{B}_k$）相似，此时网络将用该模式代表它。为了使向量 $\boldsymbol{T}_k$、$\boldsymbol{B}_k$ 更好地代表 $\boldsymbol{X}$ 的特点，将按如下方法对它们进行修改：

对 $1 \leqslant i \leqslant n$，令

$$b_{ik} = \frac{L c_i}{L - 1 + \sum_{j=1}^{n} c_j} \qquad 8-16$$

$$t_{ki} = c_i \qquad 8-17$$

与式 8－15 类似，8－16 中的 L 也是一个常数。由式 8－7、8－8 可知，矩阵 T 的任意元素的值只可能从 1 变成 0，而不可能从 0 变成 1。所以，在 8.2 节中，要求用 1 初始化 $\boldsymbol{T}$ 的所有元素。

由式 8－5、8－7、8－8、8－17 推知，如果 RN_k 对应的模式代表类 $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_d\}$，则有

$$\boldsymbol{T}_k = \boldsymbol{X}_1 \wedge \boldsymbol{X}_2 \wedge \cdots \wedge \boldsymbol{X}_d \qquad 8-18$$

这就是说，网络将向量共有的东西作为它的类表示，这也符合一般意义下的"共同特征"的要求。

式 8－16 中的 $\sum_{j=1}^{n} c_j$ 是非常重要的，不妨将它看成向量 $\boldsymbol{C}$ 的一个度量，它越大，产生的权值就越小；它越小，产生的权值就越大。这使得当一个向量是另一个向量的子集时，能够获得较好的操作。仍取

$$\boldsymbol{X}_1 = (1,0,0,0,0)$$

$$\boldsymbol{X}_2 = (1,0,0,1,1)$$

$$\boldsymbol{X}_3 = (1,0,0,1,0)$$

设 $\boldsymbol{X}_1$、$\boldsymbol{X}_2$ 分别使 RN_1、RN_2 激发。对应地，设 $\boldsymbol{T}_1 = \boldsymbol{X}_1$、$\boldsymbol{T}_2 = \boldsymbol{X}_2$。如果式 8－16 中没有 $\sum_{j=1}^{n} c_j$，则此时 $\boldsymbol{B}_1 = \boldsymbol{T}_1$、$\boldsymbol{B}_2 = \boldsymbol{T}_2$。那么，当 $\boldsymbol{X}_1$ 再一次被输入时，RN_1、RN_2 因为获得的网络输入相同而都有被选中的可能。如果 RN_2 被选中，则会导致网络运行错误，使得原有的分类被严重破坏：

(1) $\boldsymbol{X}_1$ 被再次输入，导致 RN_2 被选中；

(2) 识别层将 $\boldsymbol{T}_2$ 送入比较层：$\boldsymbol{P} = \boldsymbol{T}_2$；

(3) 此时，$\boldsymbol{C} = \boldsymbol{P} \wedge \boldsymbol{X}_1 = \boldsymbol{X}_1$；

(4) $\boldsymbol{C}$ 与 $\boldsymbol{X}_1$ 被送入系统复位控制模块，计算出 $s = 1$；

(5) 因为 $s > \rho$，所以对网络进行训练：$\boldsymbol{T}_2 = \boldsymbol{C}$。

显然，其原值被破坏了。而当选择一个适当的 L，同时在调整 $\boldsymbol{B}$ 时保留 $\sum_{j=1}^{n} c_j$，这个问题就可以避免了。

需要提醒读者注意的是，网络的分类并不是一成不变的，为说明此问题，继续使用上面例子中的输入向量，这里，取 $L = 6$，用式 8－15 对 $\boldsymbol{B}$ 进行初始化，使得 $\boldsymbol{B}$ 的所有元素均取值 0.6。

(1) $\boldsymbol{X}_1$ 的输入导致 RN_1 被激发；$\boldsymbol{B}_1$ 被训练后取值为(1,0,0,0,0)。

(2) 输入 $\boldsymbol{X}_2$ 时，RN_1、RN_2 所获得的网络输入分别为 1 和 1.8，这导致 $RN2$ 被激发；$\boldsymbol{B}_2$ 被训练后取值为(0.6,0,0,0.6,0.6)。

(3) 此时，如果 $\boldsymbol{X}_1$ 再次被输入，RN_1、RN_2 所获得的网络输入分别为 1 和 0.6，从而正确的神经元被激发；如果 $\boldsymbol{X}_2$ 再次被输入，RN_1、RN_2 所获得的网络输入分别为 1 和 1.8，从而也仍然有正确的神经元被激发。

(4) 当 $\boldsymbol{X}_3$ 被输入时，RN_1、RN_2 所获得的网络输入分别为 1 和 1.2，从而 RN_2 被激发，此时，$\boldsymbol{T}_2=(1,0,0,1,1)$被送入比较层，使得 $\boldsymbol{C} = \boldsymbol{T}_2 \wedge \boldsymbol{X}_3 = \boldsymbol{X}_3$，从而导致 $s = 1 > \rho$。

(5) 网络进入训练。$\boldsymbol{T}_2$、$\boldsymbol{B}_2$ 被修改：

$$\boldsymbol{T}_2 = (1,0,0,1,0)$$

$$\boldsymbol{B}_2 = (6/7,0,0,6/7,0)$$

(6) 当再次输入 $\boldsymbol{X}_2$ 时，RN_1、RN_2 所获得的网络输入分别为 1 和 12/7，这再次导致 RN_2 被激发。但是，此时识别层送给比较层的 $\boldsymbol{T}_2=(1,0,0,1,0)$，从而有 $s = 2/3$，如果系统的复位控制参数 $\rho > 2/3$，此时系统会重新为 $\boldsymbol{X}_3$ 选择一个新的神经元。

由此例可见，网络是根据当前存储的模式确定将一个输入向量划归某一类的。随着网络的运行，它还可以根据新的情况进行重新分类。所以，当一个输入向量 $\boldsymbol{X}$ 在某一时期被归入某一类后，当它再次出现时，有可能被分到另一类中。

最后需要指出的是，虽然 ART 网络因为具有可塑性，使得它可以边工作边学习。但是，如果能一次性地收集到网络运行将会遇到的所有情况的代表向量，也可以让网络先进行训练，在训练完成后，再投入运行。这是因为，ART 在训练稳定之后，任意一个具有与训练向量基本特点相同的输入向量都会在被输入时“立即”激发它所对应的神经元，无需再进行查找。并且，对任意的输入向量序列，网络在经过有限次的学习后，将产生一个稳定的权向量集合，任何重新出现的训练向量序列都不会导致 ART 网络联接权的不断变化。在这种情况下，由于运行中无需再考虑训练的问题，这会使得其运行期间的效率大大提高。同时这也提醒我们，网络按要求进入训练期的时候，当它正好激发原来激发的神经元时，相应的训练工作有可能是可以省略的。

练 习 题

1. 什么叫网络的可塑性？BP 网、对传网等为什么不具有可塑性？

2. 可塑性网络应该具有哪几种特殊功能？它们在保证网络的可塑性方面各起什么作用？

3. 图 8－2 指出了网络为了保证可塑性，当它遇到一个输入向量时应该采取的策略。分类和类表示功能是这种网络最基本的功能。请你指出 ART 是如何分类，如何表示一个类的。

4. 请根据图 8－3 和图 8－4 叙述 ART 的工作过程，并说明 ART 是如何在保证网络的稳定性的前提下、又使网络具有了可塑性的。

5. 叙述比较层和识别层的工作原理。

6. 如何理解 ART 是稳定的，同时，ART 又允许网络在运行过程中根据当前的运行情况改变一个输入向量的分类的现象。也就说，在 ART 中，可能出现这种现象：一个输入向量可能在某一次输入时被划归一类，而在其后的再次输入时却被划归到了另一类。这种现象与 ART 的稳定性是否矛盾？为什么？

7. 根据 ART 的性能，请指出该模型所适应的范围。

8. 如果将本章介绍的基本 ART 模型扩展到"对一个输入向量，识别层可以用激发不同的神经元的组合来表示其所在的类"的情况，那么，ART 的工作过程应该做哪些相应的改动？

9. 已知在三维空间中的一个给定范围内非均匀地分布着 m 个点，试用 ART 实现对这些点的自动分类。假定点与点之间的距离用欧几里德距离进行度量，并要求按距离的远近进行分类：距离较近者分为一类，距离远者分入另一类。三维空间中的点用(X,Y,Z)表示。

10. 请将练习题 9 扩展到 m 维空间上。

11. ART 模型是 Grossberg 及其助手们经过一段时间的研究积累提出来的，在你了解到此模型的工作原理后，请你考虑他们研究工作的思路是什么样的。

参考文献

1 Philip D. Wasserman. Neural computing: theory and practice. Van Nostrand Reinh*old*, 1989

2 Patrick K. Simpson. Artificial neural systems - foundation, paradigms, applications, and implement. Pergamon Press Inc, 1990

3 Frank C. Hoppensteadt. An introduction to the mathematics of neurons. Cambridge Univ Press, 1986

4 胡守仁,余少波,戴葵. 神经网络导论.第1版,长沙:国防科技大学出版社,1993

5 杨行峻,郑君里. 人工神经网络.第1版,北京:高等教育出版社,1992

6 胡守仁,沈清,胡德文等. 神经网络应用技术. 第1版,长沙:国防科技大学出版社,1993